
Ministério da
Ciência, Tecnologia
e Inovação


sid.inpe.br/mtc-m19/2011/12.06.19.01-TDI

# SpaceESB: UM AMBIENTE COLABORATIVO PARA APOIO AO PROJETO CONCEITUAL DE SATÉLITES USANDO BARRAMENTO CORPORATIVO DE SERVIÇOS

Ariana Cristina Caetano de Souza

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Walter Abrahão dos Santos, aprovada em 21 de dezembro de 2011.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3ATRJUH>

INPE
São José dos Campos
2011

**PUBLICADO POR:**

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@sid.inpe.br


Ministério da
Ciência, Tecnologia
e Inovação


sid.inpe.br/mtc-m19/2011/12.06.19.01-TDI

# SpaceESB: UM AMBIENTE COLABORATIVO PARA APOIO AO PROJETO CONCEITUAL DE SATÉLITES USANDO BARRAMENTO CORPORATIVO DE SERVIÇOS

Ariana Cristina Caetano de Souza

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais, orientada pelo Dr. Walter Abrahão dos Santos, aprovada em 21 de dezembro de 2011.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3ATRJUH>

INPE
São José dos Campos
2011

Dados Internacionais de Catalogação na Publicação (CIP)

So89s

Souza, Ariana Cristina Caetano de.
SpaceESB: um ambiente colaborativo para apoio ao projeto conceitual de satélites usando barramento corporativo de serviços / Ariana Cristina Caetano de Souza. – São José dos Campos : INPE, 2011.
xx + 85 p. ; (sid.inpe.br/mtc-m19/2011/12.06.19.01-TDI)

Dissertação (Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2012.
Orientador : Dr. Walter Abrahão dos Santos.

1. Arquitetura Orientada a Serviços (SOA). 2. Barramento Corporativo de Serviços (ESB). 3. ambiente colaborativo de engenharia de sistemas. 4 projeto conceitual de satélites. I.Título.

CDU 004.75

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de Mestre em

Engenharia e Tecnologia Espaciais/Gerenciamento de Sistemas Espaciais

Dr. Germano de Souza Kienbaum

Presidente / INPE / SJCampos - SP

Dr. Walter Abrahão dos Santos

Orientador(a) / INPE / São José dos Campos - SP

Dr. Álvaro Augusto Neto

Convidado(a) / ITA / São José dos Campos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

Aluno (a): Ariana Cristina Caetano de Souza

São José dos Campos, 21 de dezembro de 2011

*A meus pais, Vagner e Cirene, a minha irmã, Ariele, e a meu noivo, Douglas, pelo apoio, compreensão e palavras de incentivo recebidos durante toda a pós-graduação.*

## AGRADECIMENTOS

Ao Instituto Nacional de Pesquisas Espaciais – INPE, pela oportunidade de estudar e utilizar suas instalações dentro de um ambiente agradável e favorável para a realização de um bom trabalho.

Aos professores do Programa de Pós-Graduação em Engenharia e Tecnologia Espacial do INPE pelo conhecimento compartilhado, especialmente, por meio das disciplinas ministradas.

Ao Professor Dr. Walter Abrahão, orientador desta dissertação, por todo empenho, sabedoria, compreensão e, acima de tudo, exigência. Gostaria de ratificar a sua competência, participação com discussões, correções, revisões e sugestões que fizeram com que concluísse este trabalho.

Não posso deixar de agradecer aos meus pais, pela educação base para minha vida e apoio nos meus estudos, itens fundamentais durante todo o percurso para chegar até aqui.

A minha irmã Ariele por todo o apoio, carinho e alegria que nunca me deixaram desanimar diante dos momentos de dificuldade.

Em especial tenho que agradecer ao Douglas, companheiro, amigo e futuro marido, que me apoiou nos bons e nos maus momentos, suportou as minhas faltas de atenção para com ele e ajudou dentro do que lhe era possível.

Agradeço também a minha grande amiga Tatiane, minha corretora ortográfica, por toda sua ajuda e por sempre acreditar em mim.

À Banca pelas valiosas sugestões e trabalho dedicado a avaliação do presente trabalho.

E acima de todos, agradeço a Deus por iluminar meu caminho e me dar forças para seguir sempre em frente.

Finalmente, a todos aqueles que de alguma forma contribuíram para esta dissertação tornar-se realidade, por todo o amor, carinho, pela confiança em mim depositada, pela ajuda, motivação e pela companhia, o meu MUITO OBRIGADA.

## RESUMO

A complexidade e demanda de sistemas espaciais é cada vez maior e tornam os aspectos técnicos relacionados ao projeto conceitual mais importantes e difíceis. Além disso, devido à pressão sobre os prazos e custos, muitos projetos espaciais demandam sua distribuição, na execução. Isto faz com que o seu desenvolvimento tenha uma divisão entre vários parceiros de projeto que, muitas vezes, estão separados temporal e geograficamente. Este cenário distribuído dificulta a troca de informações e aumenta os riscos de concepção. A fase conceitual de projetos de satélite é uma fase chave que mapeia as necessidades dos clientes em funções do produto e é onde a arquitetura funcional (e às vezes a arquitetura física) é decidida. Quando mal executada, a partir da fase conceitual, vários problemas podem surgir. Para lidar com alguns dos aspectos de projeto distribuído de sistemas espaciais, este trabalho propõe um ambiente colaborativo de apoio á engenharia de sistemas, chamado SpaceESB, mostrando como alguns processos da fase conceitual de uma missão espacial podem ser disponibilizados com base no reuso e integração da informação. Desta forma pode-se tratar melhor a natureza interdisciplinar do domínio, através do paradigma da Arquitetura Orientada a Serviços e do emprego de um Barramento Corporativo de Serviços (ESB) onde aplicações tornam-se distribuídas e a sua reutilização promovida.

**SpaceESB: A COLLABORATIVE ENVIRONMENT TO SUPPORT THE CONCEPTUAL DESIGN OF SPACECRAFTS USING AN ENTERPRISE SERVICE BUS**

## ABSTRACT

The complexity and demand of space systems are growing and making the technical aspects related to the conceptual design more important and difficult. In addition, due to pressure on time and costs, many projects require spatial distribution. This makes their developments to be divided among various project partners, sometimes temporary and geographically separated, making information exchange difficult and increasing risks. The conceptual design is a key phase that maps client needs to product functions. This is where functional architecture (and sometimes technical architecture) is being decided. Various problems may come up if decisions are wrongly taken.. To deal with some aspects of the distributed design of space systems, this work proposes a collaborative environment to support systems engineering, called SpaceESB, showing how some processes of the conceptual phase of a space mission can be made available based on the re-use and integration information. Therefore it allows one to deal with the interdisciplinary nature of the domain, through the Service-Orientated Architecture and the employment of an Enterprise Service Bus, so that applications become distributed and their re-use promoted.

## LISTA DE FIGURAS

**Pág.**

## LISTA DE TABELAS

**Pág.**

## LISTA DE SIGLAS E ABREVIATURAS

| | |
|---|---|
| ACDH | *Attitude Control and On-Board Data Handling* |
| API | *Application Programming Interface* |
| B2B | *Business-to-Business* |
| BPEL | *Business Process Execution Language* |
| CASA | *Composite Application Service Assembly* |
| CORBA | *Common Object Request Broker Architecture* |
| CRM | *Customer Relationship Management* |
| DCOM | *Distributed Component Object Model* |
| DCS | *Data Collection System* |
| EAI | *Enterprise Application Integration* |
| ERP | *Enterprise Resource Planning* |
| ESA | *European Space Agency* |
| ESB | *Enterprise Service Bus* |
| FTP | *File Transfer Protocol* |
| GPS | *Global Positioning System* |
| HTTP | *Hypertext Transfer Protocol* |
| IDE | *Integrated Development Environment* |
| INPE | Instituto Nacional de Pesquisas Espaciais |
| IP | *Internet Protocol* |
| ITA | Instituto Tecnológico de Aeronáutica |
| JavaME | *Java Micro Edition* |
| JavaRMI | *Java Remote Method Invocation* |
| JBI | *Java Business Integration* |
| JMS | *Java Message Service* |
| kW | *Kilowatt* |
| MEMS | *MicroElectro-Mechanical-Systems* |
| NASA/JPL | *National Aeronautics and Space Administration/Jet Propulsion Laboratory* |
| OASIS | *Organization for the Advancement of Structured Information Standards* |
| PDC | *Project Design Center* |
| PDF | *Portable Document Format* |
| PSS | *Power Supply Subsystem* |
| QoS | *Quality of Service* |
| REST | *Representational State Transfer* |
| RPC | *Remote Procedure Call* |
| SFTP | *Simple File Transfer Protocol* |
| SLA | *Service Level Agreement* |
| SMTP | *Simple Mail Transfer Protocol* |
| SOA | *Service Oriented Architecture* |
| SOAP | *Simple Object Access Protocol* |
| SysML | *Systems Modeling Language* |

| | |
|---|---|
| TI | Tecnologia da Informação |
| TMTC | *Telemetry and Telecommand* |
| UDDI | *Universal Description Discovery and Integration* |
| URL | *Universal Resource Locator* |
| W | *Watt* |
| W3C | *World Wide Web Consortium* |
| WS | *Web Service* |
| WSDL | *Web Service Definition Language* |
| XLS | *eXceL Spreadsheet* |
| XMI | *XML Metadata Interchange* |
| XML | *eXtensible Markup Language* |
| XSLT | *eXtensible Stylesheet Language for Transformation* |

## SUMÁRIO

**Pág.**

# 1 INTRODUÇÃO

No início da corrida espacial, na década de 50, os sistemas espaciais como os satélites, por exemplo, eram apenas um recurso para uso militar. Porém, nos dias de hoje, os sistemas espaciais são usados com diversas finalidades, sendo de suma importância para a era atual, uma era de informações, transmissões e comunicações.

Devido a isso, a demanda gerada por clientes de serviços providos por sistemas espaciais é crescente, em contrapartida os prazos e custos para o desenvolvimento desses sistemas é cada vez menor (BANDECCHI et al., 1999).

Esse aumento de demanda, com requisitos cada vez mais exigentes, tem aumentado a complexidade do desenvolvimento de sistemas espaciais e consequentemente tem exigido maiores esforços em uma das fases chave do ciclo de vida de um sistema espacial: a fase de projeto conceitual (WILKE et al., 2000).

O projeto conceitual é uma atividade interdisciplinar que envolve várias linhas de raciocínio e grande quantidade de informação. As decisões tomadas durante essa fase têm grande impacto na qualidade do produto final, bem como nos seus custos e prazos (ALMEIDA, 2000). Isso tem destacado o quão importante é a fase de projeto conceitual de projeto e exigido uma melhora constante dos processos dessa fase, a fim de atender a essas necessidades.

## 1.1. Motivação

A fase de projeto conceitual é uma das fases iniciais do ciclo de vida do projeto. Nela as necessidades do cliente são mapeadas para uma arquitetura funcional e algumas vezes, até mesmo para uma arquitetura física.

As atividades de engenharia dos sistemas espaciais costumam ser extremamente complexas, pois as decisões tomadas na elaboração da

arquitetura de um projeto têm uma natureza altamente acoplada. Isso implica que as definições e evoluções de cada componente de um sistema espacial têm um impacto sobre outros componentes e que qualquer mudança se propagará por todo o sistema (BANDECCHI et al, 1999).

Durante a fase de desenvolvimento conceitual de um programa espacial, há vários níveis de análise necessários para se definir os requisitos que guiarão o processo posterior de aquisições de subsistemas. A cada nível de análise, um determinado grau de abrangência é necessário para garantir que o entendimento do sistema como um todo seja mantido. Ao mesmo tempo, um grau adequado de profundidade na análise é também exercitado visando responder as questões sem sobrecarregar a análise com detalhes e complexidades desnecessárias.

Neste contexto, o projeto conceitual de sistemas espaciais é definido como a primeira tentativa de se definir características detalhadas de subsistemas de um satélite bem como seu desempenho, custo, etc (AGUILAR; DAWDY; LAW, 1998).Desta forma, o processo de projeto conceitual gerará como saída as especificações bem como o detalhamento da análise, projeto e verificação de subsistemas sem que o produto final, o satélite, seja ainda construído.

O conhecimento e o trabalho conjunto de engenheiros das diversas disciplinas envolvidas no projeto são requeridos pela fase de projeto conceitual. Porém, essa cooperação entre as equipes multidisciplinares pode se tornar difícil quando os seus membros estão em locais diferentes, e torna-se ainda mais se o satélite é uma colaboração entre diversas nações. Fatores como diferenças de fuso horário, as barreiras linguísticas, numeração e unidades de medida, e diferentes plataformas de TI podem ter um efeito adverso dificultando a comunicação e gerando uma demora na tomada de decisões. Decisões essas que são cruciais para que o projeto atinja melhores resultados e menores custos e prazos.

Para ajudar os engenheiros em suas tomadas de decisões, que podem determinar o sucesso ou falha de uma missão, ferramentas e metodologias de engenharia apropriadas para o desenvolvimento são necessárias para que as análises sejam realizadas de forma rápida e precisa. Iniciativas de uso da abordagem e de ferramentas colaborativas e concorrentes para projetar, analisar e avaliar completa e rapidamente os projetos conceituais de missões espaciais já é uma realidade (MEIJERS et al., 2004), por exemplo, no *Project Design Center* (PDC) da *National Aeronautics and Space Administration/Jet Propulsion Laboratory* (NASA/JPL) (WALL, 2000).

O avanço das aplicações em rede e o interesse em ambientes colaborativos para promover a engenharia de sistemas em nuvens (*cloud-enabled*), têm sido objeto de estudo atual (ROSS et al., 2006). A engenharia de sistemas espaciais demanda esse tipo de ambiente assim como requer conhecimentos multidisciplinares que vão desde o computador de bordo até a interface com o veículo lançador.

Na *European Space Agency* (ESA) abordagens de engenharia simultânea em um ambiente colaborativo têm sido bem sucedidas (BANDECCHI et al., 1999). Com várias instalações de engenharia simultânea as organizações do setor espacial europeu perceberam a necessidade de colaborar e de poder trocar dados do estudo entre as instalações. Uma vez que cada unidade tem suas ferramentas, plataformas e sistemas específicos, a troca de dados só pode ser bem sucedida se for baseada em um protocolo aberto e neutro para o intercâmbio de dados (ECSS E-TM-10-25, 2010).

Por isso, foi estabelecida pela ESA, uma infraestrutura reutilizável usando equipamentos e ferramentas de engenharia já existentes e já empregadas na avaliação de missões espaciais. A partir de um modelo de dados padrão e do uso de *web services* todas essas ferramentas puderam ser integradas e as informações sobre o projeto puderam ser disponibilizadas de forma centralizada e acessível a todos os envolvidos no projeto.

No Instituto Nacional de Pesquisas Espaciais (INPE), a Engenharia de Sistemas Espaciais utiliza um conjunto de ferramental baseado em planilhas manualmente geradas além de um conjunto de aplicativos específicos para cada subsistema de satélites que não trabalham integradamente. Dados são trocados de maneira ainda manual e sequencial não explorando o caráter paralelo e distribuído de decisões de projeto conceitual (LEONOR, 2010).

A falta de ferramentas e de um ambiente que permita a interoperabilidade de sistemas e o intercâmbio das informações da fase de projeto conceitual de satélites foi o motivador para esse trabalho.

Este cenário altamente acoplado e distribuído pode ser resolvido graças à disponibilidade de padrões abertos para reduzir as barreiras entre diferentes plataformas, bem como infraestrutura para apoiar a oferta de serviços. Essa abstração é possível através de SOA e via *web services* (ERL, 2004) que estão sendo adotados tornando processos de negócio mais eficientes e eficazes. Essas tecnologias contribuem para modelar processos de negócios, criar soluções, projetos, desenvolver e prestar serviços.

## 1.2. Objetivo

Atualmente no INPE o projeto conceitual é realizado por um grupo restrito de arquitetos e engenheiros de sistemas a partir de requisitos e restrições de missões oriundas de demandas da sociedade brasileira e constantes do PNAE (Plano Nacional de Atividades Espaciais). Basicamente, o grupo de Engenharia de Sistemas utiliza planilha de cálculo em seus balanços técnicos e softwares dedicados a um determinado subsistema espacial. Através de interações entre o grupo, uma arquitetura de satélite começa a maturar até que a mesma seja homologada por uma revisão de projeto.

Nos casos onde o desenvolvimento do satélite é realizado através do estabelecimento de parcerias com outras empresas, essa interação e troca de informação entre os grupos de engenheiros envolvidos no projeto se torna

ainda mais difícil, já que todas as interações têm um baixo grau de automação onde se carece de uma visão do sistema como um todo fazendo com que o processo seja lento.

Com base na motivação apresentada na Seção 1.1 e no cenário atual vivido pelo INPE este trabalho tem como objetivo contribuir com a fase de projeto conceitual de sistemas espaciais tornando-a mais eficiente.

Através da criaçãode um ambiente colaborativo, denominado SpaceESB, focadona fase conceitual de projetos de satélites e que utiliza a *Service Oriented Architecture* ou Arquitetura Orientada a Serviços (SOA) e o conceito de *Enterprise Service Bus* ou Barramento Corporativo de Serviços (ESB) pretende-se disponibilizar um conjunto simples de serviços que realiza balanços no satélite (número de termistores, número de comandos diretos e área do painel solar) e integrar uma ferramenta já existente que irão apoiar as decisões tomadas na fase de projeto conceitual de um satélite. Esses serviços podem ser acessados por sistemas de empresas parceiras do projeto, de qualquer lugar e independentemente de plataforma.

Isto permitirá uma melhor interação entre as disciplinas do ciclo de vida do projeto e garantirá que os impactos das decisões tomadas por uma equipe sejam rapidamente avaliados pelas demais equipes envolvidas diminuindo os riscos de concepções errôneas.

## 1.3. Metodologia

Uma revisão da literatura foi realizada para identificar importantes características da engenharia de sistemas e do projeto conceitual de satélites. Muitas discussões podem ser encontradas no domínio público, permitindo a definição do problema e seus fatores contribuintes.

A revisão da literatura também inclui a conceituação de SOA, ESB e web services e a identificação das características que viabilizam seu uso, como: flexibilidade, reutilização e integração.

O próximo passo foi identificar as ferramentas existentes no mercado que permitiriam a criação do ambiente colaborativo. O critério para a opção foram ferramentas gratuitas e que oferecessem funcionalidades que permitiriam a implantação de SOA de forma rápida e fácil.

Após esta etapa foi o momento da implementação do SpaceESB, onde as ferramentas base para a criação do ambiente colaborativo foram instaladas e configuradas, os serviços foram desenvolvidos em linguagem de programação e implantados no ambiente. Por fim, foi realizada a integração de uma ferramenta já existente ao SpaceESB para testar a todo o ambiente.

A etapa final foi documentar a implementação e demonstrar a utilização do SpaceESB. A partir dos resultados atingidos, as conclusões foram tiradas sobre a eficácia global da proposta para apoiar a fase de projeto conceitual de satélites e recomendações para futuros trabalhos foram feitas.

## 1.4. Organização do texto

Esse primeiro Capítulo corresponde à introdução do trabalho onde o tema da dissertação é contextualizado.

O segundo Capítulo refere-se à fundamentação teórica onde são apresentados conceitos e definições sobre o ciclo de vida e a fase de projeto conceitual de projetos espaciais, a Arquitetura Orientada a Serviços, as tecnologias usadas em *web services* e ESB e uma revisão de trabalhos anteriores.

No Capítulo três é apresentada a proposta deste trabalho, descrevendo o conceito de SpaceESB, as ferramentas utilizadas, a forma de desenvolvimento, integração e implantação. Na parte final do capítulo é descrito o funcionamento de todo o ambiente construído.

No Capítulo quatro é feito um estudo de caso demonstrando a utilização e como o SpaceESB pode apoiar os engenheiros de projeto em sua tomada de decisão.

Finalmente, o Capítulo 5, encerra o trabalho com as considerações finais e propostas de futuros trabalhos.

# 2 FUNDAMENTAÇÃO TEÓRICA

Conforme as organizações crescem, estas desenvolvem um número cada vez maior de aplicações distribuídas por uma série de departamentos e lugares. Ao mesmo tempo, está se tornando cada vez mais importante para as organizações o compartilhamento eficiente das informações entre esses aplicativos (ROSS et al., 2006).

A Arquitetura Orientada a Serviços foi proposta para suprir essa necessidade de permitir a intercomunicação entre as aplicações. Em SOA é necessário o desenvolvimento de interfaces de baixo acoplamento para aplicações (serviços) que podem ser usadas para troca de informações entre aplicações. Ao combinar esses serviços, é possível desenvolver aplicações *adhoc* (*mash-ups*) conforme a necessidade das organizações.

Neste capítulo serão discutidos aspectos relativos ao projeto conceitual de satélites, às características de SOA, bem como de *web services* e do ESB. Ao final, é apresentada uma revisão dos trabalhos que apoiaram o desenvolvimento deste trabalho.

## 2.1. Projeto Conceitual

Projetos espaciais são geralmente estruturados em várias etapas, refletindo os diferentes tipos de atividades exercidas dentro do projeto. Todo o processo é chamado de ciclo de vida de um sistema espacial para conceber, projetar, construir e operar um sistema espacial (BERTRAND, 1999). A Figura 2.1 mostra a nomenclatura usada na *European Space Agency* (ESA) para as fases do ciclo de vida dos projetos (ECSS-M-30-01A, 1999).

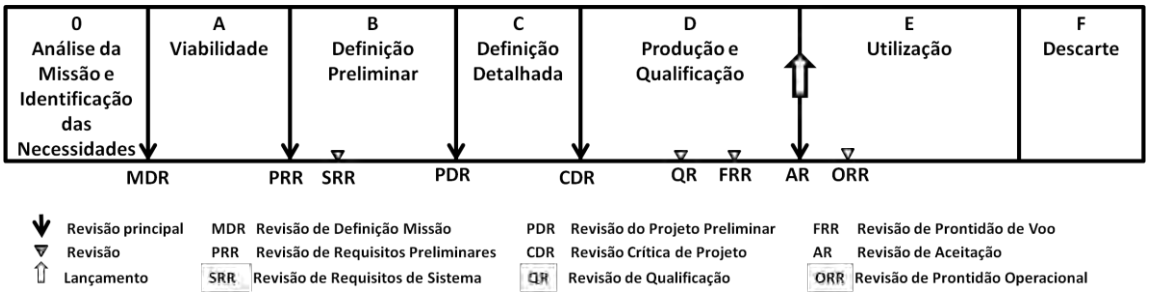



Figura 2.1 - Ciclo de vida dos projetos espaciais
Fonte: BERTRAND (1999)

- Fase 0, Análise da Missão e Identificação das Necessidades - nesta primeira fase do projeto, a tarefa é identificar as necessidades ou interesses dos *stakeholders* que serão traduzidos em um conjunto de requisitos e restrições do sistema.

- Fase A, Viabilidade - esta fase é responsável por realizar uma análise de viabilidade, todos os conceitos retidos na Fase 0 devem ser analisados e avaliados em seus aspectos técnicos, econômicos e organizacionais.

- Fase B, Definição Preliminar - as atividades desta fase têm como objetivo refinar o que foi definido nas fases anteriores e dividir os requisitos do sistema em subsistemas.

- Fase C, Definição Detalhada - durante esta fase e criado um projeto detalhado do sistema, modelos de engenharia são construídos e testados, componentes críticos são desenvolvidos e pré-qualificados e planos de montagem, testes e qualificação são desenvolvidos.

- Fase D, Produção e Qualificação - inclui a produção, integração e verificação do sistema.

- Fase E, Utilização - corresponde à fase em que o sistema espacial é implantado e utilizado. Para a ESA, o lançamento do segmento espacial marca o início da utilização do sistema.

- Fase F, Descarte - durante esta fase o plano de descarte, elaborado na Fase B e refinado até a Fase E, é executado. Uma estratégia de descarte inclui uma transferência para uma órbita cemitério ou transferência de volta para a Terra, destruição no momento da reentrada, entre outros.

O termo projeto conceitual é aplicado a todas as atividades relacionadas à definição e avaliação dos conceitos de um sistema, e corresponde as Fases 0 e A do ciclo de vida de um projeto.

A fase conceitual de projeto é uma das primeiras de uma sequência de passos que transformam ideias em um produto. Conforme mostra a Figura 2.2, em uma visão resumida, nessa fase chave existem 3 processos macro com alguns subprocessos que devem ser executados (LARSON; WERTZ, 1991).

O primeiro processo é chamado de definição dos objetivos da missão, nesse processo são definidos os objetivos gerais que o sistema deve alcançar para ser produtivo e, posteriormente, os objetivos definidos são tranformados em requisitos funcionais, restrições e requisitos operacionais. Esse primeiro processo é formado pelos subprocessos: objetivos da missão e definição dos requisitos e restrições preliminares.

O segundo processo, caracterização da missão, é composto pelos subprocessos: identificar conceitos aplicáveis à missão, definir arquiteturas alternativas da missão e system level trades. A caracterização da missão caracterização da missão é o processo inicial de seleção e definição de uma missão espacial. O objetivo é selecionar a melhor abordagem da vasta gama disponível para executar uma missão espacial.

O terceiro, e último processo, é a avaliação e seleção dos conceitos da missão formado por quatro subprocessos: avaliar conceitos, gerar medidas de efetividade, documento base e conceitos base selecionados para estudos aprofundados.

Nessa fase, avalia-se a capacidade das arquiteturas alternativas, definidas no processo de caracterização da missão, em cumprir os objetivos fundamentais da missão de acordo com as medidas de efetividade definidas para o processo. Em seguida, a arquitetura selecionada é documentada e seus conceitos serão estudados de forma mais profunda nas próximas fases de desenvolvimento do sistema espacial.

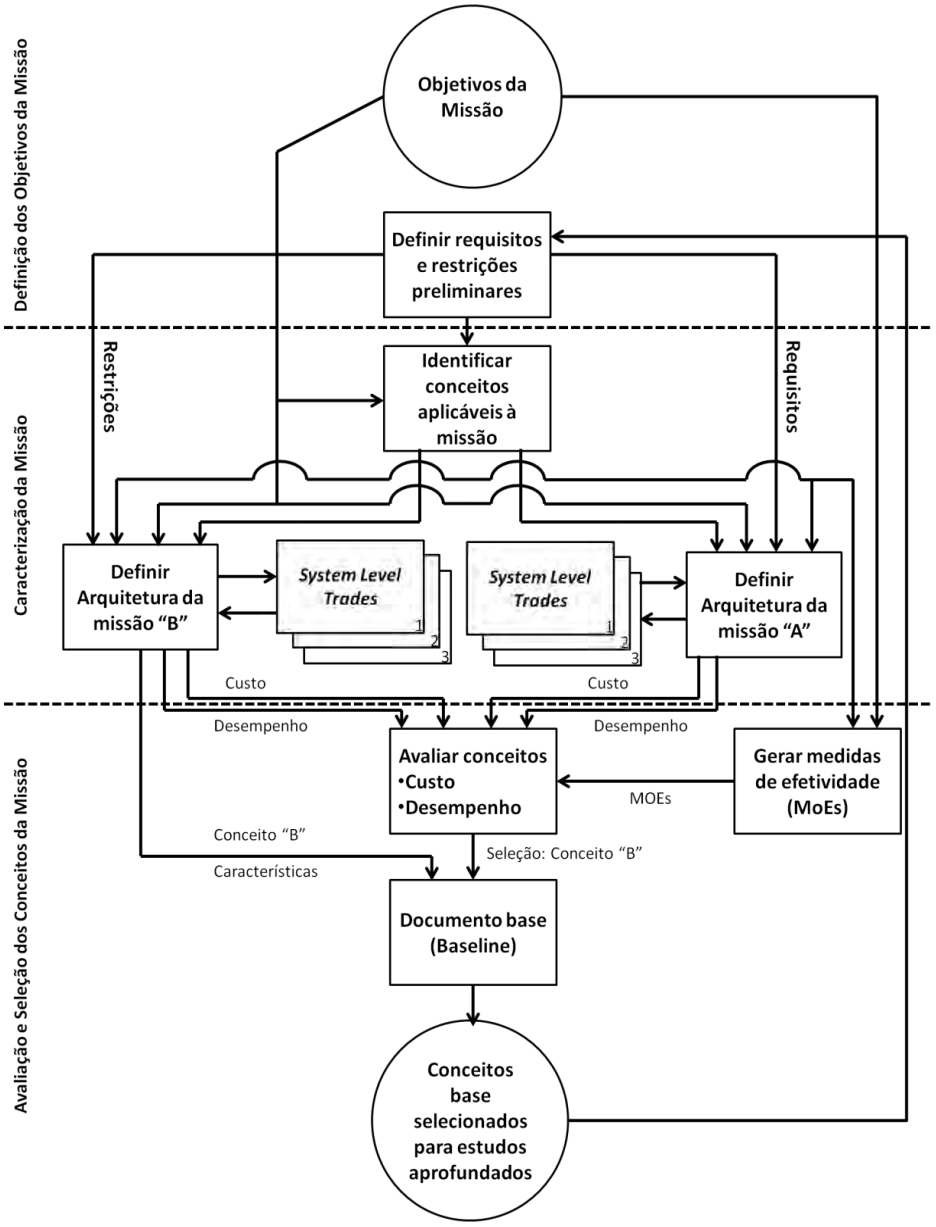



Figura 2.2 - Fluxo do Projeto Conceitual
Fonte: LARSON; WERTZ (1991).

Dessa forma, o fluxo de execução do projeto conceitual é o seguinte: nos primeiros processos os objetivos da missão, suas restrições e seus requisitos são definidos, Em seguida, possíveis arquiteturas da missão são geradas tendo como base: restrições, requisitos, conceitos e objetivos da missão. Essas opções são então exploradas em detalhes, sua viabilidade em termos de custo e desempenho é avaliada, e a arquitetura mais adequada é selecionada. Daí em diante o projeto pode progredir para o próximo estágio de seu ciclo de vida.

Neste trabalho alguns serviços serão alocados para apoiar algumas atividades no projeto conceitual de satélites.

## 2.2. Arquitetura Orientada a Serviços

Ao longo de muitos anos, as empresas investem cada vez mais no desenvolvimento de soluções de TI. Essas soluções têm plataformas, linguagens de programação, paradigmas de programação e até *middleware* diferentes, fazendo com que haja uma grande redundância de funcionalidades e alto acoplamento entre as soluções.

Devido a esse crescimento desordenado, essas empresas possuem inúmeras aplicações legadas que não podem ser descartadas e necessitam ser integradas às novas soluções.

Ao contrário do que muitos pensam, a Arquitetura Orientada a Serviços não foi inventada. SOA é uma evolução natural da arquitetura de sistemas tradicional em resposta às demandas de um mercado que exibe, a cada dia, mais agilidade e qualidade (ERL, 2004).

O modelo arquitetônico SOA acomoda inúmeras visões e consolida qualidades oriundas da arquitetura dos objetos distribuídos e componentes, como: distribuição, reutilização, contratos, estilo modular, baixo acoplamento e conectividade (LAZZERI, 2010).

Segundo a OASIS, 2006, SOA é um paradigma para organizar e utilizar competências distribuídas que podem estar sob controle de diferentes domínios proprietários, é um meio para organizar as soluções que promovem o reuso, crescimento e interoperabilidade.

A Arquitetura Orientada a Serviços “é um modelo que descreve a disposição e os princípios que regem os elementos para a provisão e o intercâmbio de serviços entre entidades provedoras e requisitantes.” (CAVALCANTI; ALMEIDA, 2010).Apesar de existirem diversas definições para SOA, todas concordam que SOA estabelece um modelo arquitetônico que visa aumentar a eficiência, agilidade e produtividade de uma empresa mediante o posicionamento de serviços como o principal meio através do qual a solução lógica é representada em apoio à realização dos objetivos estratégicos associados à computação orientada para serviços (ERL, 2011). Em SOA as funcionalidades do negócio são disponibilizadas na forma de serviços possibilitando maior reuso, redução de redundâncias e maior eficiência na manutenção. Um serviço em SOA deve ter os seguintes atributos (HOLLEY; ARSANJANI, 2010):

- Sem estado (*Stateless*): os *web services* não dependem do contexto ou do estado de outros serviços. Um serviço *stateless* não guarda o estado e trata cada requisição como uma transação nova e independente, sem relação com requisições anteriores ou futuras;

- Detectável: um *web service* deve ser descoberto por potenciais consumidores do serviço. Serviços são publicados em um repositório pelos provedores de serviços, assim aqueles podem ser detectados e invocados;

- Autodescrição: um *web service* deve ter um contrato ou interface bem definido, descrevendo as informações que um consumidor de serviços necessita para descobrir e conectar-se ao serviço;

- Combinável: um *web service* pode compor e/ou ser composto por outros serviços para criar novas soluções;

- Baixo acoplamento: é a capacidade de um serviço ser independente de outros serviços para realizar uma tarefa;

- Regido por políticas: Serviços são construídos por contrato. Relações entre os serviços são regidas por políticas e acordos de nível de serviço (*Service Level Agreement* - SLA), promovendo a consistência do processo e reduzindo a complexidade;

- Localização, linguagem e protocolos independentes: serviços são projetados para serem transparentes, sendo acessíveis por qualquer usuário, em qualquer plataforma, de qualquer lugar.

Além disso, os serviços em uma Arquitetura Orientada a Serviços têm as seguintes características:

- Granularidade: serviços, geralmente, têm granularidade grossa. A granularidade está relacionada com o grau de profundidade sobre algum nível de abstração, ou seja, a granularidade do serviço é o grau de modularidade dos serviços ou o escopo da funcionalidade exposto por um serviço. Um serviço de granularidade grossa possui grande capacidade funcional, já um serviço com granularidade fina possui pequenas funcionalidades;

- Assíncrono: a comunicação assíncrona não é uma característica obrigatória de um serviço, mas aumenta a escalabilidade das soluções. Um comportamento assíncrono permite que um consumidor faça uma requisição a um serviço e continue seu processamento até que o provedor de serviço retorne uma resposta.

Como uma forma de arquitetura tecnológica, uma implementação de SOA pode consistir de uma combinação de tecnologias, produtos, APIs, suportando

extensões de infraestrutura, e várias outras peças. A face atual de um serviço implantado segundo SOA é única dentro de cada empresa; porém, é caracterizada pela introdução de novas tecnologias e plataformas que apoiam especificamente a criação, execução e evolução de soluções orientadas a serviços (ERL, 2011).

O uso de SOA muda a maneira como as soluções são desenvolvidas e implementadas, a exposição das funcionalidades do negócio como serviços possibilita às empresas respostas mais rápidas às mudanças do mercado. SOA permite também que os sistemas já existentes sejam combinados com novos esforços de desenvolvimento (aplicativos compostos). Daí a motivação em empregá-la como base para alguns processos de engenharia de sistemas.

Frequentemente, na Arquitetura Orientada a Serviços, os serviços são organizados através de um "barramento de serviços" que disponibiliza interfaces, ou contratos, acessíveis através de *web services* ou outra forma de comunicação entre aplicações.

A arquitetura SOA, mostrada genericamente na Figura 2.3, é baseada nos princípios da computação distribuída e utiliza o paradigma *request / reply* para estabelecer a comunicação entre os sistemas clientes e os sistemas que implementam os serviços (WU et al., 2008).

Neste cenário, o provedor de serviços publica e registra um serviço em um repositório. O consumidor de serviços utiliza o registro para obter o contrato do serviço, que contém informações sobre as operações disponíveis e os parâmetros que o serviço necessita para se comunicar. Através desse contrato o consumidor inicia as requisições ao serviço.

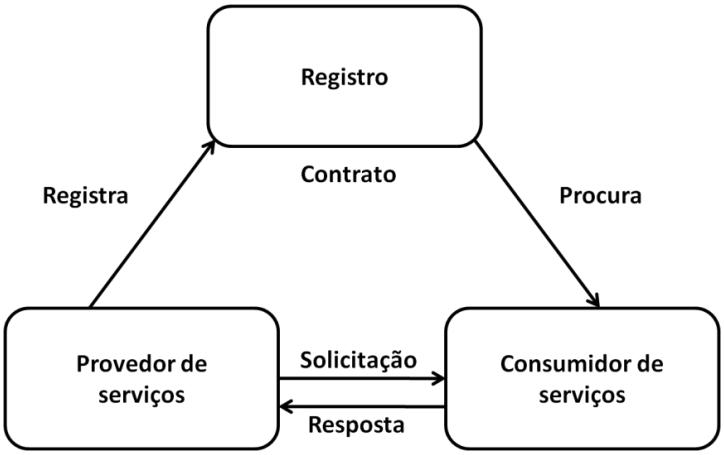



Figura 2.3 - Esquema básico do ambiente SOA

Fonte: adaptada de W3C (2002).

### 2.2.1. Fundamentos da Arquitetura Orientada a Serviços

A Engenharia de Software, no decorrer dos anos, evoluiu da programação estruturada para a orientada a objetos, para a baseada em componentes, e por fim para a orientada a serviços. Cada uma das evoluções se baseia na abstração anterior e SOA abrange as melhores práticas da orientação a objetos e componentes. A Figura 2.4 ilustra os diferentes níveis de abstração: de objetos a serviços (HOLLEY; ARSANJANI, 2010).

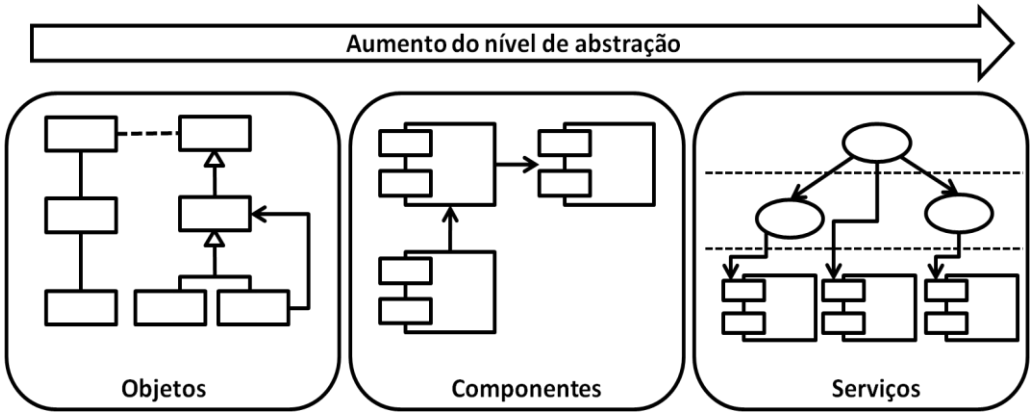



Figura 2.4 - Níveis de abstração

Fonte: adaptada de Holley e Arsanjani, (2010).

Conforme mostra a Figura 2.5, SOA tem como base os seguintes aspectos: interface com consumidor, processos de negócio, serviços, componentes de serviço, sistemas operacionais e integração.

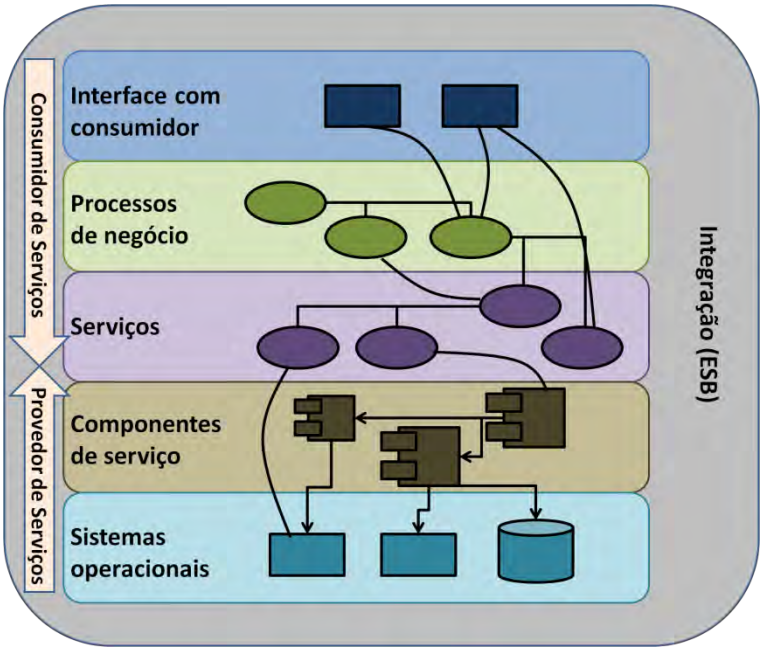



Figura 2.5 - Camadas lógicas da Arquitetura SOA
Fonte: adaptada de Open Group, (2009).

Na Figura 2.5 os aspectos de SOA são representados em camadas lógicas que podem ser usadas como diretrizes para o projeto de soluções orientadas a serviços.

A camada de consumidor, ou a camada de apresentação, fornece os recursos necessários para entregar as funções de TI e dados para usuários finais. Essa camada também pode fornecer uma interface de comunicação entre as aplicações. A camada de consumidor oferece a capacidade de criar rapidamente o *front-end* de processos de negócios e aplicações compostas para responder às mudanças nas necessidades do usuário através *mash-ups*, portais e outros mecanismos.

A camada de processos representa o fluxo de atividades requeridas para completar um processo de negócio. Nesta camada são definidas as

composições, coreografias e orquestrações dos serviços, ou seja, a camada de processos de negócios abrange a representação de processo e métodos de composição para agregar serviços fracamente acoplados como um processo de sequenciamento alinhado com os objetivos do negócio.

Os serviços são os principais elementos de uma arquitetura orientada a serviços. Na camada de serviços é onde estes são expostos, podendo ser descobertos e acessados através de suas interfaces bem definidas.

A camada de componentes é responsável pela execução de funcionalidade e manutenção da qualidade dos serviços expostos. Estes componentes são um conjunto de ativos da organização, sendo responsáveis por garantir a conformidade com SLAs através da aplicação de melhores práticas de arquitetura. Essa camada geralmente usa tecnologias, tais como servidores de aplicativos, para implementar os componentes, gerenciamento de carga, alta disponibilidade e balanceamento de carga.

A camada operacional inclui os sistemas legados de uma organização, como *Customer Relationship Management* (CRM), *Enterprise Resource Planning* (ERP), entre outros, que estão em execução e apóiam as atividades de negócios.

Por fim, a camada de integração, geralmente um ESB é essencial em SOA, pois permite mediar, transformar, rotear e transportar as requisições entre o consumidor e o provedor de serviços.

A partir da Figura 2.5, fica claro que SOA evoluiu a partir das camadas de provedores de serviços, focadas na construção de objetos e componentes usando as abordagens orientadas a objetos e a componentes, para a integração com as camadas de consumidores de serviços, focadas na construção e orquestração de serviços para implementar os processos de negócios usando a abordagem orientada a serviços.

O modelo de camadas apresentado é apenas uma referência para Arquitetura Orientada a Serviços que permite entendimento das entidades significativas e os relacionamentos entre elas em um ambiente orientado a serviço.

## 2.3. Web Services

Em seus fundamentos, a Arquitetura Orientada a Serviços gira em torno do paradigma de serviços e, muitas vezes, SOA e *web services* são tratados como sinônimos, mas não são a mesma coisa (JOSUTTIS, 2007).

SOA, é um modelo de arquitetura agnóstico para qualquer plataforma tecnológica. Com isso, é dada a liberdade para perseguir continuamente os objetivos estratégicos associados a computação orientada a serviços, possibilitando alavancar avanços tecnológicos no futuro (ERL, 2011). Atualmente, a plataforma tecnológica mais associada com a realização de SOA são os *web services* (MORO et al., 2009).

A implementação da Arquitetura Orientada a Serviços tem sido realizada através de *web services*, pois essa tecnologia tem se consolidado como uma das mais adequadas para a integração de sistemas heterogêneos (MOREIRA, 2005). Mas a popularidade dos *web services* precede SOA. A sua utilização inicial foi principalmente dentro das soluções tradicionais de sistemas distribuídos, mais comumente utilizados para facilitar canais de integração ponto-a-ponto. À medida que a maturidade e a adoção de padrões dos *web services* aumentou, surgiu assim o âmbito da utilização da SOA (ERL, 2007).

Outros exemplos de arquiteturas que podem ser usadas para a implementação de SOA são (SOMMERVILLE, 2007): *Common Object Request Broker Architecture* (CORBA), que define uma arquitetura de objetos para a computação distribuída, *Distributed Component Object Model* (DCOM), promovido pela Microsoft e *Java Remote Method Invocation* (JavaRMI).

Segundo o W3C, 2004, *web services* são usados na integração de sistemas e na comunicação entre diferentes aplicações. É possível utilizá-los para integrar aplicações escritas em diferentes linguagens e executando em diferentes plataformas através de mensagens baseadas em *eXtensible Markup Language* (XML). As trocas de informações realizadas com *web services* utilizam um canal de comunicação, geralmente, o protocolo *Hypertext Transfer Protocol* (HTTP).

Existem alguns tipos de arquiteturas para o desenvolvimento de *web services*, entretanto o RPC, WS-* (WSDL/SOAP) e REST os mais conhecidos e usados hoje em dia.

A arquitetura WS-* é a mais utilizada hoje em dia e é composta por mais de 20 especificações, donde SOAP, WSDL e UDDI são as especificações base deste conjunto e descritas a seguir (MORO et al., 2009).

A Figura 2.6 mostra os elementos fundamentais de um *web service*, tais como o consumidor de serviço, o provedor serviço e seu relacionamento. O consumidor de serviços invoca ou usa um serviço mediante a descrição do serviço em *Web Service Definition Language* (WSDL) para obter as informações necessárias para consumir o serviço. Um *broker* é usado para encontrar um serviço publicado em *Universal Description Discovery & Integration* (UDDI). Provedores de serviços utilizam UDDI para publicar os serviços que eles oferecem. Consumidores de serviços usam UDDI para descobrir serviços que lhes interessam e obter os dados necessários para utilizar esses serviços.

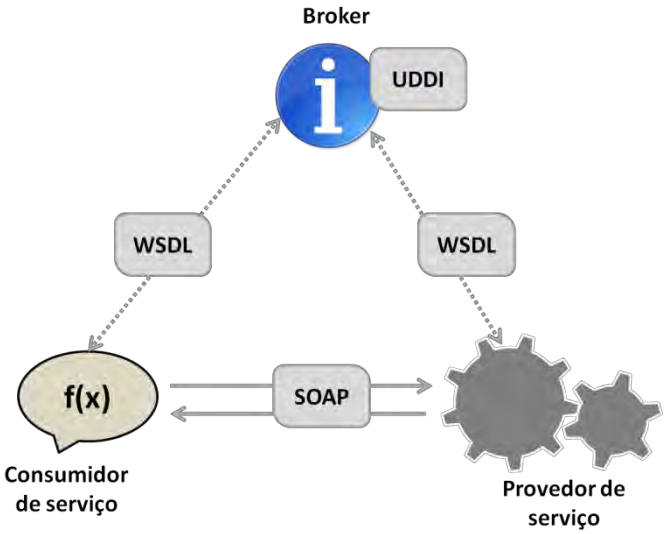



Figura 2.6 - Esquema básico de *web services* e seu sistema de mensagens
Fonte: adaptada de Gottschalk et al.(2002).

### 2.3.1. SOAP

*Simple Object Access Protocol* (SOAP) é um protocolo padrão para troca de mensagens. Como SOAP adota um formato XML, o qual é baseado em texto, SOAP é interpretado pela maioria de ambientes novos, e já existentes, e pode ser transportado sobre vários protocolos.

Na maioria das vezes SOAP é transportado sobre o protocolo HTTP, mas também já existem implementações que enviam SOAP usando *Java Message Service* (JMS).

Em *web services* uma requisição feita por um consumidor de serviços é formatada em uma mensagem XML, a qual é encapsulada utilizado o protocolo SOAP para ser transportado sobre HTTP. No provedor de serviços, essa mensagem é desempacotada, os dados são lidos e processados e o resultado é novamente encapsulado por SOAP e enviado pelo protocolo HTTP ao solicitante.

### 2.3.2. UDDI

O UDDI é um diretório de serviços que permite a descrição e descoberta de: (1) empresas, organizações e outros provedores de serviços, (2) *web services* disponíveis, e (3) as interfaces que podem ser utilizadas para acessar esses serviços. Com base em um conjunto comum de padrões da indústria, incluindo HTTP, XML, e SOAP, o UDDI fornece uma infraestrutura fundamental para um ambiente baseado em *web services* (OASIS, 2004).

### 2.3.3. WSDL

WSDL é uma linguagem de definição de interface baseada em XML que permite descrever os contratos dos serviços. Esse contrato é um documento que contém a descrição do serviço, suas operações e a forma como acessá-lo.

Os documentos WSDL são constituídos por palavras-chave como (W3C, 2001):

- *PortType*: descreve as operações existentes no web service e seus dados de entrada e saída;
- *Binding*: provê informações de como interagir com o *web service* especificando o protocolo a ser usado, como por exemplo, SOAP/HTTP;
- *Port*: descreve o endereço através do qual o *web service* poderá ser invocado, geralmente representado por uma URL.

Através do WSDL, aplicações desenvolvidas em qualquer linguagem e plataforma conseguem acessar um *web service* e suas operações. Por exemplo, a partir de WSDL de um *web service* em Java, executando em um ambiente *WebSphere*, e oferecendo invocação por meio de HTTP, um desenvolvedor .NET pode importar o WSDL e gerar um aplicativo que faça solicitações ao serviço.

## 2.4. ESB - Barramento Corporativo de Serviços

Nos últimos anos algumas tendências tecnológicas significativas, tais como *Service Oriented Architecture* (SOA), *Enterprise Application Integration* (EAI), *Business-to-Business* (B2B) e *web services* têm tentado enfrentar os desafios de melhorar os resultados e aumentar o valor da gestão integrada dos processos de negócio. O conceito de ESB agrega as melhores características dessas, e de outras, tendências e tecnologias, é uma abordagem para a integração que pode fornecer as bases para um baixo acoplamento e a integração de redes altamente distribuídas (CHAPPELL, 2004).

A implementação de SOA necessita de uma infraestrutura mais sofisticada e gerenciável que possa suportar um grande volume de interações e que se integre com infraestruturas já existentes. ESBs estão emergindo como o conceito para essa infraestrutura.

Em um modelo usando SOA, o ESB é a infraestrutura que possibilita a alta interoperabilidade entre os serviços, possibilitando a chamada aos serviços oferecidos. Em uma visão simplificada, o papel do ESB é rotear e transportar uma requisição a um serviço para o correto provedor do serviço.

O verdadeiro valor de um ESB, no entanto, é promover uma infraestrutura para SOA de forma a refletir as necessidades de uma organização, fornecendo os níveis de serviço adequados e capacidade de gerenciamento, operação e integração em um ambiente heterogêneo. As principais responsabilidades de um ESB são (JOSUTTIS, 2007):

- Transformação de dados: responsável por prover a transformação de protocolos, por exemplo, entrar em um HTTP e sair em um SFTP, para que seja possível integrar plataformas e linguagens de programação diferentes;

- Roteamento (inteligente): responsável por mandar as informações para um lugar ou para outro, isso pode ser feito de maneira estática ou dinâmica. Podendo usar roteamento baseado em regras com o Drools, por exemplo;

- Lidar com confiabilidade: provê capacidades para lidar com falhas e erros técnicos;

- Gerenciamento de serviços: responsável por gerir o ciclo de vida de serviços. Prevê o gerenciamento de diferentes níveis de operação e a definição de atributos descritivos para todos os serviços implementados. Outro aspecto importante é o registro de serviços, que permite o mapeamento dos serviços disponíveis, facilitando o monitoramento da saúde dos serviços presentes, assim como o gerenciamento centralizado de exceções;

- Monitoramento e *logging*: fornece informações sobre o estado dos serviços, das aplicações e das requisições;

- Prover conectividade: responsável integrar diferentes plataformas de *hardware* e *software*, mesmo diante de diferentes *middleware* e protocolos facilitando a interação entre o consumidor e o provedor de serviços;

- Lidar com segurança: suporta capacidades de segurança, como autenticação, autorização, confidencialidade e normas de segurança.

A Figura 2.7, mostra o esquema de ESB, cuja estrutura típica é composta por três camadas:

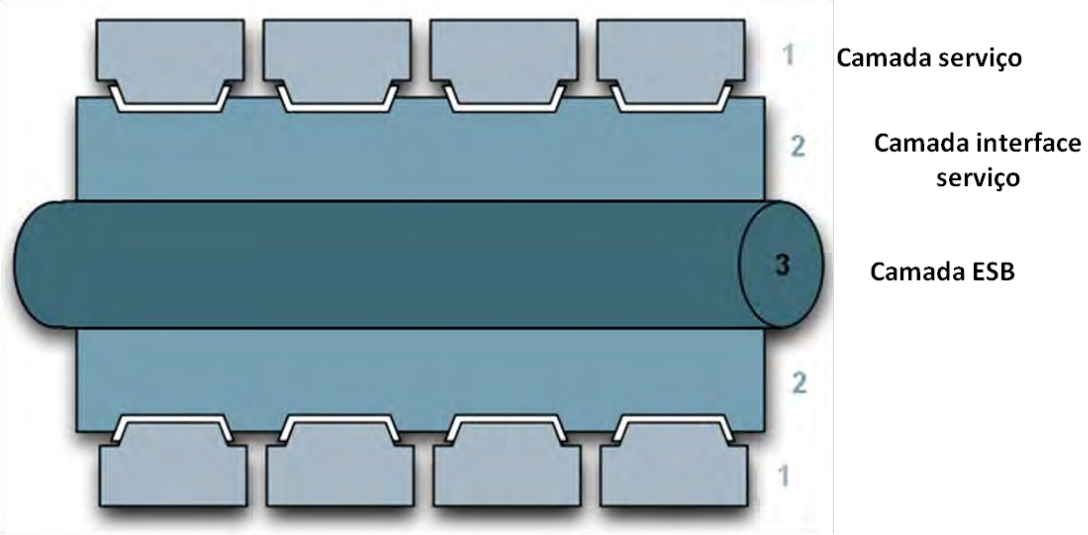



Figura 2.7 - Estrutura de um ESB
Fonte: LONGO (2009).

- Camada de serviço: Trata-se dos serviços, ou seja, da exposição de funcionalidades e/ou regras de negócio de sistemas existentes;

- Camada de interface do serviço: Representa a interação dos serviços com o barramento para colaboração com os demais serviços ligados ao barramento. Quando se fala de *web services*, essa camada poderia ser, por exemplo, SOAP ou REST;

- Camada ESB: É o barramento de serviços ou ESB, que faz a mediação da comunicação entre os serviços.

## 2.5. Trabalhos Anteriores

Várias iniciativas para ambientes de colaboração dentro da área de engenharia de sistemas espaciais têm sido publicadas, como em Watkins et al., 2007, Bandecchi et al., 1999, Dos Santos et al., 2009.

A Tabela 2.1, lista um sumário dos principais trabalhos anteriores examinados e focados no domínio de ambientes colaborativos para engenharia de sistemas espaciais.

Tabela 2.1 - Trabalhos anteriores na linha de colaboração em engenharia de sistemas espaciais.

| Abordagem / Trabalho | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ambiente distribuído colaborativo | ✓ | ✓ | ✓ | | ✓ | | ✓ | | | | | | |
| Web services | | | | | | | | | | | | ✓ | ✓ |
| Ferramenta de apoio à engenharia de sistemas espaciais | | | | ✓ | | | | | ✓ | | ✓ | | |
| Engenharia Concorrente | | | | ✓ | | | | ✓ | | ✓ | ✓ | | ✓ |
| Integração de sistemas legados | | | | ✓ | | | | | ✓ | | | ✓ | ✓ |
| Metodologias Colaborativas e Concorrentes | | | | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | | | |
| Engenharia Dirigida a Modelos | | | | | | | | | ✓ | | ✓ | | |
| Fase Projeto Conceitual | | | | | | ✓ | | | | ✓ | ✓ | ✓ | ✓ |

| | | | | | |
|---|---|---|---|---|---|
| 01 | Watkins et al., 2007 | 06 | Lewis et al., 2007 | 11 | Leonor, 2010 |
| 02 | Bandecchi et al., 1999 | 07 | Aguilar, 2008 | 12 | Hardwick, 2009 |
| 03 | Santos et al., 2009 | 08 | Morse, 2006 | 13 | ECSS E-TM-10-25A, 2010 |
| 04 | Santos, 1997 | 09 | Wilke et al., 2000 | | |
| 05 | Rodriguez; Alashaab, 2005 | 10 | Smith et al., 2001 | | |

Um framework colaborativo para apoiar a engenharia de sistemas espaciais é apresentada em Dos Santos, 1997. Uma arquitetura de sistema web baseada em conhecimento e para desenvolvimento colaborativo de produtos é descrita em Rodriguez; Alashaab, 2005. Usando engenharia simultânea, modelos alternativos são utilizados em Lewis et al., 2007 na fase de projeto conceitual de satélites.

O trabalho de Aguilar, 2008, mostra um ambiente de desenvolvimento de sistemas concorrentes onde engenheiros e clientes trabalham juntos para desenvolver novos sistemas espaciais. Modelos para a próxima geração de Engenharia Simultânea são apresentados em Morse, 2006.

Em Wilke et al., 2000, é apresentada uma ferramenta chamada MuSSat para apoiar as fases iniciais de projeto de satélites comerciais que melhora a qualidade dos dados e reduz o tempo de desenvolvimento. A ferramenta permite a modelagem do sistema, realiza análise de custos do projeto e permite estimar custos para variações de MOE's

Em Smith et al., 2001, são apresentadas metodologias colaborativas e concorrentes para projetar, analisar e avaliar completa e rapidamente os projetos conceituais de missões espaciais empregadas no Team-X do JPL.

Uma ferramenta de software baseada em conhecimento para apoio no projeto de satélites é apresentada em Leonor, 2010. O uso de web services como abordagem para a criação de ambientes colaborativos de engenharia é descrito em Hardwick, 2009 servindo de inspiração para a criação do SpaceESB.

A ferramenta criada por Leonor, 2010, será integrada ao SpaceESB, neste trabalho, para demonstrar como SOA facilita a integração de ferramentas legadas. Já o trabalho de Hardwick, 2009, foi utilizado como base para a extensão dos serviços disponibilizados no SpaceESB.

Finalmente, em ECSS E-TM-10-25A, 2010 é descrito um modelo para engenharia simultânea em ambientes distribuídos e a necessidade de colaboração entre os mesmos. Nele são utilizados *web services*, construídos em .NET para realizar a integração de várias ferramentas já utilizadas pela ESA. A abordagem utilizada permite a transferência de dados apenas entre instalações compatíveis que utilizam sempre a mesma plataforma.

# 3 DESENVOLVIMENTO DO PROTÓTIPO SpaceESB

Neste capítulo serão descritos o processo de desenvolvimento do SpaceESB, a integração e as ferramentas utilizadas neste trabalho.

O SpaceESB é um protótipo de ambiente colaborativo que utiliza a Arquitetura Orientada a Serviços e um Barramento Corporativo de Serviços para disponibilizar e integrar processos e aplicações da fase conceitual de projeto de satélites. Isso permite que empresas parceiras de projeto possam consumir esses serviços independentemente de plataformas.

Optou-se por usar SOA e ESB no projeto, pois esta arquitetura pode criar um ambiente flexível e reutilizável. Enquanto o ESB é focado na interconexão dos sistemas, mais especificamente nos protocolos, plataformas e formas de acesso, SOA se foca sobre contratos e reutilização de sistemas (KEEN et al., 2004).

Neste ambiente distribuído, denominado SpaceESB, foram disponibilizados três diferentes serviços que geram balanços para um projeto de satélite: número de termistores, número de comandos diretos e área do painel solar. Ao SpaceESB, também foi integrado um sistema legado, desenvolvido anteriormente pelo INPE, chamado SatBudgets. Como o foco do trabalho é a criação do ambiente distribuído, não foi feita grande expansão da lista de balanços disponíveis e sim no seu encapsulamento em serviço.

O SatBudgets é uma ferramenta que auxilia na fase conceitual de um projeto de satélite gerando, inicialmente, balanços mecânico e elétrico (LEONOR, 2010). Com a integração do SpaceESB ao SatBudgets as funcionalidades da ferramenta foram estendidas, pois além dos balanços mecânico e elétrico o SatBudgets passou a disponibilizar os balanços de número de termistores, comandos diretos e área do painel solar através da oferta dos serviços disponibilizados.O SpaceESB em conjunto com a ferramenta SatBudgets implementa algumas das atividades pertencentes aos processos da fase de

projeto conceitual, tipicamente, "*System Level Trades*" e Avaliação de Conceitos mostrados na Figura 2.2 do capítulo 2.

A Figura 3.1 mostra o esquema proposto para o SpaceESB. Nesse ambiente os sistemas das empresas parceiras envolvidas em um projeto se conectam ao SpaceESB para acessar as atividades da fase conceitual de projetos disponíveis como serviços. Os sistemas dos parceiros não têm conhecimento sobre o destino dos serviços e todas as chamadas aos serviços são enviadas pelo próprio ESB responsabilizando-se por rotear as mensagens e tratar as exceções. Dessa forma, cria-se um desacoplamento entre o provedor e o consumidor e mudanças no código-fonte dos serviços podem ser realizadas sem afetar o fluxo de trabalho.

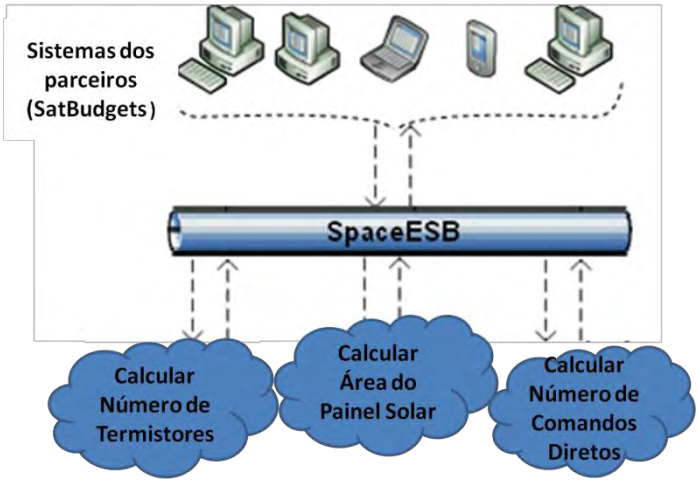



Figura 3.1 - Esquema do SpaceESB

## 3.1. Ferramentas Utilizadas

A materialização de SOA, assim como qualquer outra implementação de software, necessita de um ambiente de desenvolvimento e de ferramentas para criar a modelagem dos negócios, os serviços, enfim, todos os elementos que farão com que SOA deixe de ser apenas teoria e seja colocado em prática.

O SpaceESB foi desenvolvido fazendo uso somente de ferramentas e *Application Programming Interface (*APIs) gratuitas. Os *web services* criados também seguiram esse requisito e foram desenvolvidos na linguagem Java.

### 3.1.1. OpenESB

O OpenESB é um projeto open-source que implementa um Enterprise Service Bus tendo como base o JBI *runtime*. O *Java Business Integration* (JBI) é uma especificação desenvolvida para a implementação de SOA, fornecendo uma arquitetura plugável de componentes e um modelo de troca de mensagens entre os componentes (OPENESB, 2011). A Figura 3.2 mostra a arquitetura do OpenESB e seus principais componentes, que serão detalhadas a seguir.

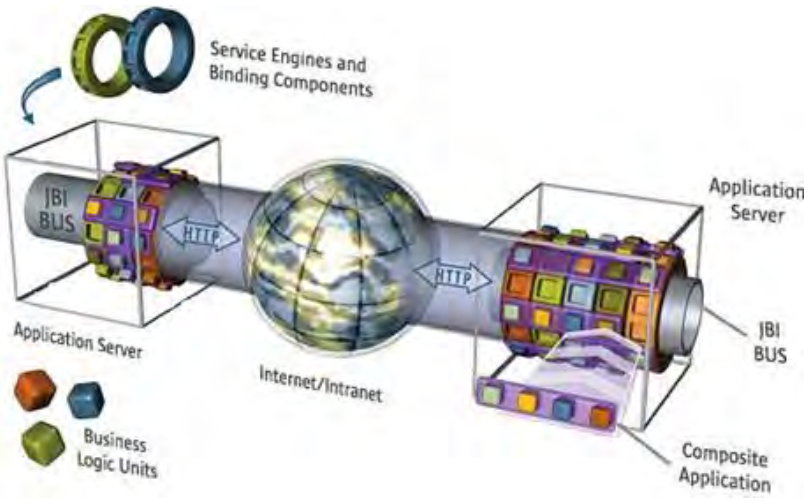



Figura 3.2 - Arquitetura do OpenESB
Fonte: adaptada de Shin, (2007).

- JBI Bus: é o barramento onde os componentes são plugados e interligados. Por ele os componentes se comunicam a partir de mensagens padronizadas.

- *Service Engine*: um componente que expõe os serviços que implementam lógicas do negócio. Um exemplo desse componente é o BPEL *Service Engine* responsável pela execução dos processos de

negócios no formato *Business Process Execution Language* (BPEL) (BPEL, 2010).

- *Binding Component*: atuam como um tradutor de protocolos de comunicação diferentes, convertendo as mensagens recebidas de serviços externos para um formato normalizado. Existem diferentes *Binding Components* para diferentes modos de interação com serviços externos, por exemplo, HTTP, FTP, SMTP, entre outros.

- *Composite Application*: é uma aplicação construída a partir da combinação de outras funções e/ou aplicações. Usando o OpenESB é possível disponibilizar *Composite Applications* combinando, por exemplo funções de um *Enterprise Resource Planning* (ERP) (SOMMERVILLE, 2007) ou serviços legados a partir apenas de sua orquestração, sem a necessidade de reescrever as funções.

O OpenESB contém vários componentes (*binding components* e *service engines*) que podem ser necessários em um cenário de integração e, quando necessário, novos componentes podem ser desenvolvidos e adicionados ao OpenESB.

OpenESB, por ser baseado em padrões abertos pode ser implantado em diversos servidores de aplicação, como JBoss, GlassFish, WebSphere, entre outros (OPENESB, 2011).

### 3.1.2. GlassFish ESB

No mercado estão disponíveis ESBs de diferentes fornecedores como: IBM WebSphere ESB, Sonic ESB, TIBCO BusinessWorks, Cape Clear, Oracle ESB. Na lista de ESBs *open source* podem ser citados, Mule ESB, Apache ServiceMix, JBoss ESB, e GlassFish ESB. Para este trabalho o ESB escolhido foi o GlassFish ESB, por estar já integrado ao IDE NetBeans, também gratuito.

O GlassFish ESB é uma distribuição binária dos componentes do OpenESB (SOMEKH, 2008). O GlassFish ESB, como seu nome sugere, vem com o *GlassFish Server*, um servidor de aplicações de código aberto que fornece o ambiente adequado para a hospedagem dos serviços, tais como segurança e gerenciamento de transações.

### 3.1.3. NetBeans IDE

O NetBeans IDE é um ambiente de desenvolvimento integrado, gratuito, que auxilia no desenvolvimento de aplicações em diferentes linguagens como Java, C++, PHP, Ruby, entre outras (NETBEANS, 2010). Como o NetBeans é escrito em Java, é independente de plataforma e funciona em qualquer sistema operacional, com uma máquina virtual Java.

O NetBeans IDE disponibiliza diversos recursos SOA, inclusive o GlassFish ESB, ferramentas gráficas para a criação dos serviços, bem como sua orquestração utilizando o módulo BPEL e a combinação de múltiplos serviços através do editor *Composite Application Service Assembly* (CASA). Além dessas ferramentas disponíveis para desenvolver aplicações SOA destacam-se também:

- Editor gráfico de arquivos WSDL;
- Ferramentas para a criação, modificação e visualização de XML Schema;
- XSLT Designer para a criação e edição de documentos XSLT;
- Ferramentas para desenvolvimento de aplicações móveis *Java Micro Edition* (JavaME);
- Ferramenta de autocompletar, facilitando a escrita do código;
- Ferramentas de debug, entre outras.

A Figura 3.3 apresenta a tela principal do IDE NetBeans com trecho de código da implementação dos *web services*.

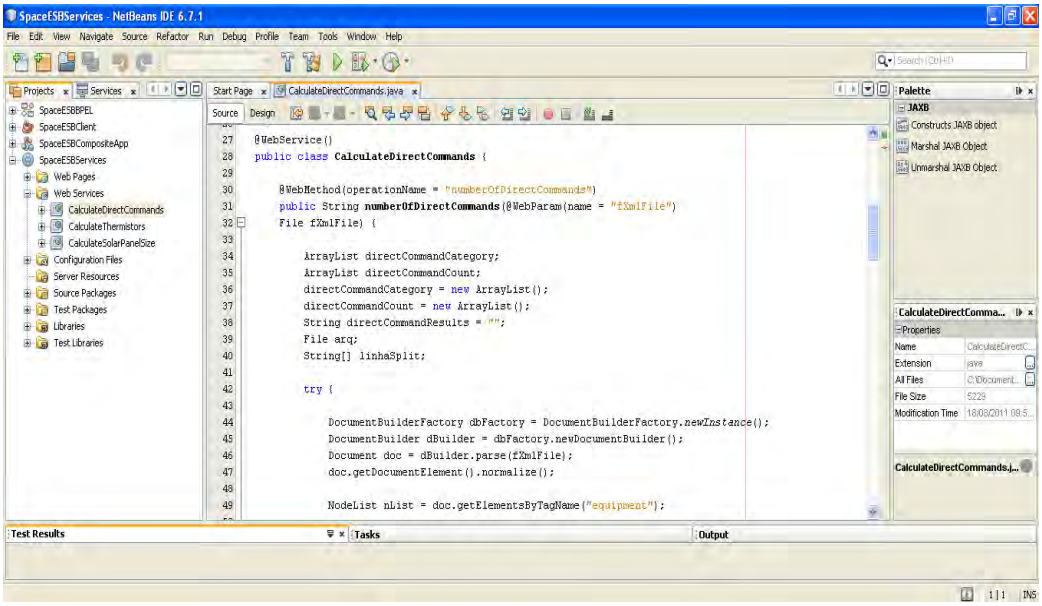



Figura 3.3 - NetBeans IDE

## 3.2. Processos do Projeto Conceitual de Satélites

Para a criação do SpaceESB, o primeiro passo é a identificação de quais atividades da fase conceitual de um projeto serão automatizadas sendo expostas como serviços.

Normalmente, em um projeto conceitual de satélite, busca-se uma arquitetura do sistema adequada que satisfaça os requisitos da missão (LARSON; WERTZ, 1991). Como uma simples ilustração de atividades desta fase inicial, este trabalho apresenta os balanços necessários para avaliar a quantidade de termistores, o número de comandos diretos para funcionalidades críticas de bordo e, a área do painel solar.

Durante o projeto conceitual de sistemas, balanços são executados iterativamente e visa dimensionar características fundamentais de projeto como massa, custo, confiabilidade, consumo, etc. Estes estão ligados geralmente as

restrições do sistema comumente denominados de “envelopes” e devem ser obedecidos para garantir a viabilidade técnica de qualquer arquitetura candidata.

Os balanços aqui tratados são relativos diretamente à disciplina de Computação de Bordo. Durante o processo de desdobramento do projeto conceitual, um engenheiro de sistemas dedicado a esta disciplina deverá ter estimativas de custo e desempenho necessário para o sistema de computação de bordo de forma que possa gerar as especificações técnicas deste subsistema a serem tratadas nas fases seguintes do projeto. Para tal, o engenheiro deve trabalhar com alguns números que quantifiquem o susbsistema como o dimensionamento do número de interfaces necessárias para leitura de canais de termistores e hardware para decodificação de comandos diretos. Naturalmente, métricas como massa e potência são presentes em todas disciplinas.

Brevemente discutidos nas seções posteriores, cada uma destes balanços é implementado como um serviço baseado em sistemas de regras de negócio de engenharia. Um conjunto extremamente simplificado de regras de negócios de (HARDWICK, 2009) foram usadas para programar a lógica dos *web services*. As melhorias realizadas nas regras criadas por Hardwick (2009) são descritas no Anexo A.

### 3.2.1. Balanço do Número de Termistores

O subsistema de controle térmico geralmente é responsável por manter todos os equipamentos a bordo de um satélite dentro de uma faixa de temperatura adequada ao seu funcionamento.

Existem duas estratégias para realizar o controle térmico: passiva e ativa (LARSON; WERTZ, 1991). Como mostrado na Figura 3.4, a estratégia passiva utiliza materiais, revestimentos, acabamentos de superfície, propriedades térmicas da estrutura, entre outros, para manter os limites de temperatura. Já a

estratégia ativa mantém os limites de temperatura através de um meio ativo, como por exemplo, aquecedores termostáticos, tubos de calor de condutância variável, sistemas de bombeamento mecânico com circuitos dotados de irradiadores e trocadores de calor (SOUZA, 2008).

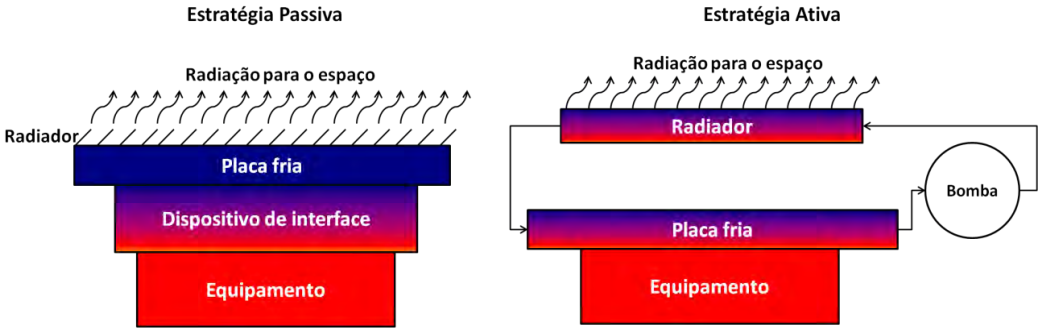



Figura 3.4 - Esquemas de controle térmico passivo e ativo
Fonte: SOUZA (2008)

Um componente importante no subsistema de Controle Térmico é o sensor de temperatura, geralmente um termistor. Durante a fase conceitual, é necessário estimar o número de termistores atribuídos para cada componente do satélite que precisam de monitoramento térmico. Esse balanço afeta o projeto de outros subsistemas acoplados, por exemplo, processamento de bordo, energia e cablagem, entre tantos.

As seguintes regras simplificadas foram aplicadas para o cálculo do número de termistores:

- Para a estrutura do satélite bastam quatro termistores para monitoração térmica;
- Se a estratégia de controle térmico usada for a ativa, todos os equipamentos (exceto o citado acima) recebem dois termistores por unidade de equipamento;

- Se a estratégia de controle térmico for passiva o número de termistores por unidade de equipamento (exceto para o listado acima) é determinado da seguinte forma:
    - Cada equipamento recebe um termistor por unidade, exceto:
        - Computadores de bordo, painéis solares e baterias recebem dois termistores por unidade;
        - Sensores solares, diplexers, híbridas e antenas não recebem termistores.

### 3.2.2. Balanço do Número de Comandos Diretos

Os comandos diretos não são processados e executados pelo computador de bordo, mas são diretamente codificados e executados. Estes são dedicados, principalmente, à execução de comandos críticos da missão, como os seguintes equipamentos: computadores de bordo, baterias e receptores de telecomunicações.

Este balanço em particular, afeta a confiabilidade do satélite e do projeto de outros subsistemas acoplados como o de processamento de bordo, comunicações e cablagem, por exemplo. As regras simplificadas aplicadas a este balanço são:

- Os seguintes equipamentos recebem dois comandos diretos por unidade de equipamento. Todos os outros equipamentos não recebem comandos diretos;
    - Computadores de bordo;
    - Bateria;
    - *Transceiver* de telecomunicação;

- Qualquer unidade de equipamento pertencente à carga útil.

### 3.2.3. Balanço da Área do Painel Solar

O subsistema de energia, esboçado na Figura 3.5 é o principal responsável pelo: (1) fornecimento de energia adquirida principalmente a partir de painéis solares; (2) armazenamento de energia através de baterias em geral, (3) condicionamento, conversão, regulação e de distribuição de energia para todos os subsistemas e equipamentos.

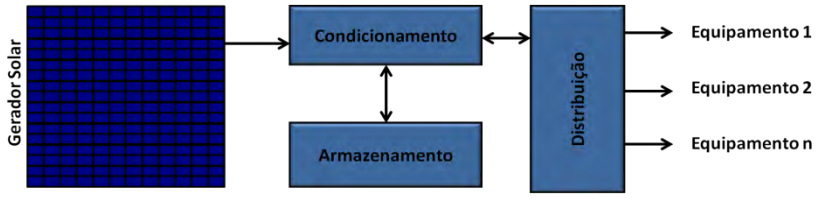



Figura 3.5 - Diagrama de blocos de um subsistema de suprimento de energia típico
Fonte: SOUZA (2008).

Para encontrar a energia demandada, a área do painel solar precisa ser avaliada e uma margem extra de segurança de geração de energia deve ser considerada. Este balanço, em particular, é vital e afeta o projeto de quase todos os subsistemas. O balanço da área do painel solar obedece às seguintes regras simplificadas:

- Para cada unidade de equipamento existe um consumo de energia estimado;
- A densidade de energia no espaço, em média, é 1.2 kW por metro quadrado;
- A área do painel solar deve fornecer 20% a mais da energia total requerida pelo satélite como margem de segurança.

## 3.3. Implementação

A primeira etapa para a implementação é a criação dos *web services,* já mencionados anteriormente. No final da criação dos *web services*, um arquivo WSDL também é gerado. O arquivo WSDL contém detalhes sobre a descrição do serviço, suas operações e as condições de acesso. A Figura 3.6 a seguir mostra um trecho do arquivo WSDL para o balanço do número de termistores.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions targetNamespace="http://service/" name="CalculateThermistorsService"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://service/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <portType name="CalculateThermistors">
    <operation name="numberOfThermistors">
      <input message="tns:numberOfThermistors"/>
      <output message="tns:numberOfThermistorsResponse"/>
    </operation>
  </portType>
  <binding name="CalculateThermistorsPortBinding" type="tns:CalculateThermistors">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="numberOfThermistors">
    </operation>
  </binding>
  <service name="CalculateThermistorsService">
    <port name="CalculateThermistorsPort"
binding="tns:CalculateThermistorsPortBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
    </port>
  </service>
</definitions>
```

Figura 3.6 - Trecho do arquivo WSDL do web service do balanço de número de termistores

A segunda etapa da implementação é realizar a orquestração dos serviços. A criação dos *web services* somente implementa as operações do serviço, mas é essencial também implementar o fluxo de execução das operações.

A orquestração é responsável por definir a sequência e condições de invocação dos serviços (BPEL, 2010). BPEL foi a estratégia escolhida para a

orquestração dos serviços que é retratada na Figura 3.7. BPEL é um dialeto executável de XML que define as interações de serviços.

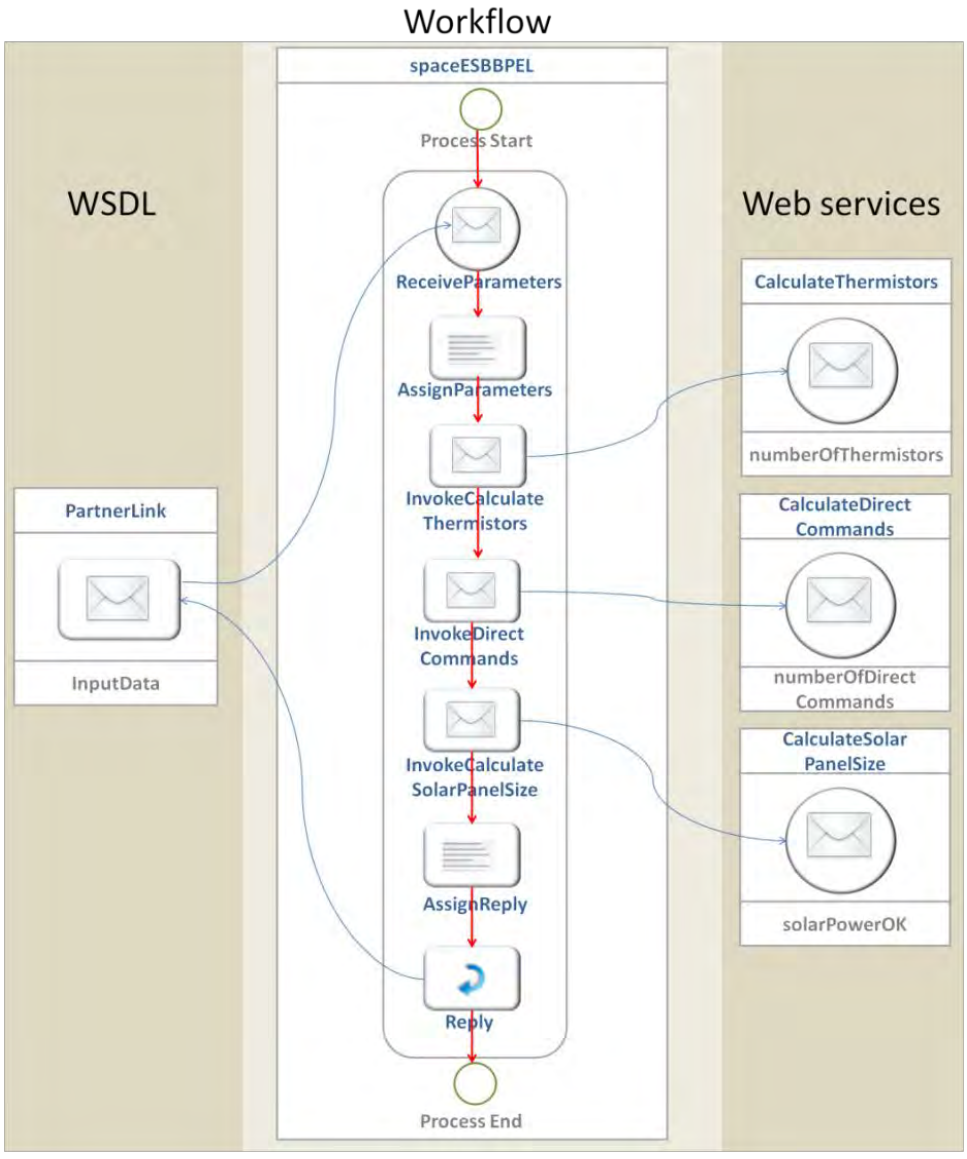



Figura 3.7 - Diagrama BPEL para os três *web services* de balanço

No lado direito da Figura 3.7 encontram-se os serviços responsáveis pela execução dos cálculos dos balanços, enquanto no lado esquerdo, o arquivo WSDL requerente. A parte central mostra o fluxo de trabalho obedecido no consumo dos três serviços. Neste caso, o fluxo de execução é bastante simples: após a entrada de dados a partir do arquivo WSDL requerente, todos

os serviços são chamados e após a sua conclusão, os resultados são enviados de volta para o solicitante.

O passo seguinte é a criação da aplicação composta, uma aplicação construída a partir da combinação dos múltiplos serviços. Ela consiste de funcionalidades extraídas de várias fontes dentro de uma Arquitetura Orientada a Serviços (SOA). Os componentes podem ser *web services* individuais, funções selecionadas a partir de outras aplicações ou sistemas inteiros cujas saídas foram empacotadas como *web services* (muitas vezes, sistemas legados). A Figura 3.8 mostra a aplicação composta criada, que contém o módulo BPEL, os serviços e como os serviços serão expostos. Usando os recursos do NetBeans, para a criação da aplicação composta, também é possível definir alguns atributos para *Quality of Service* (QoS), como qual ação será tomada em caso de falha, número de tentativas de acesso antes de usar a opção de falha, entre outras.

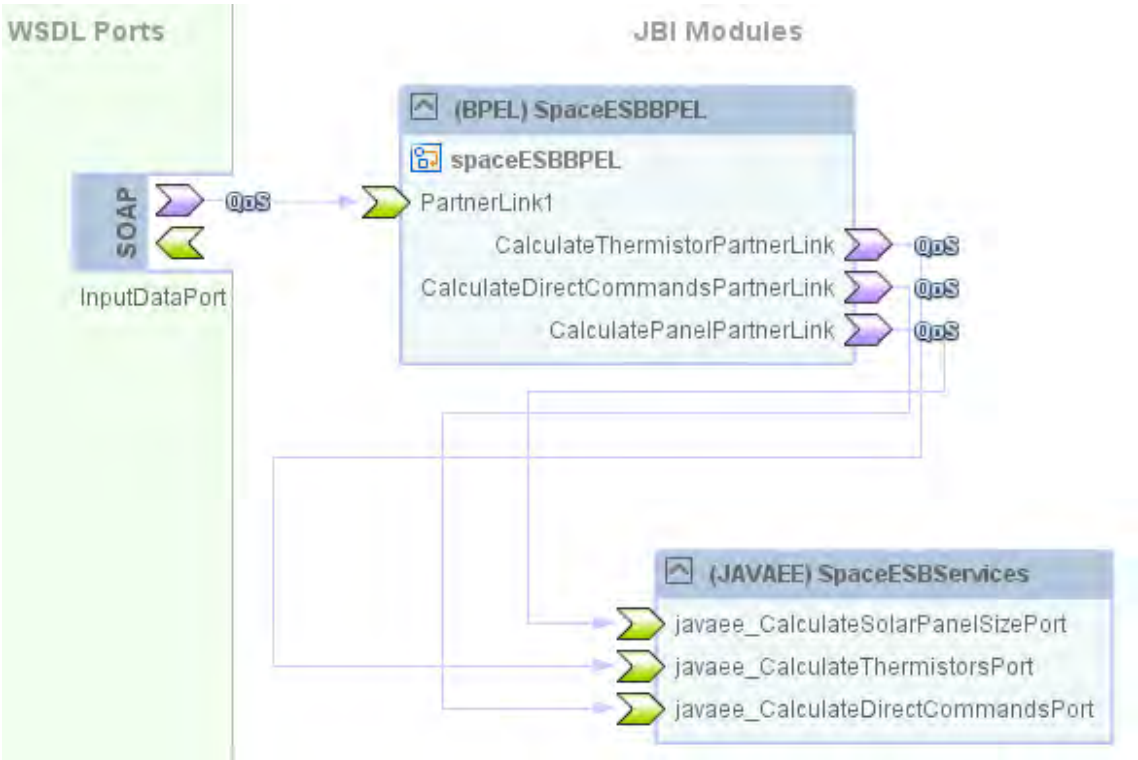



Figura 3.8 - Aplicação Composta para a exposição dos *web services.*

A última fase da implementação é a realização da implantação da aplicação composta no servidor ESB, tornando-a disponível para todos os parceiros do projeto que possam precisar das funcionalidades.

### 3.4. Integração SatBudgets - SpaceESB

Para testar o funcionamento do ambiente distribuído, foi integrado ao SatBudgets o SpaceESB.

O SatBudgets é uma ferramenta que auxilia na fase conceitual de um projeto de satélite gerando automaticamente, a partir da modelagem SysML, balanços mecânico e elétrico para mostrar a viabilidade de construção do sistema ou a necessidade de alterações no projeto do satélite (LEONOR, 2010).

A Figura 3.9 mostra o esquema de integração e execução entre o SatBudgets e o SpaceESB.

O SatBudgets realiza a leitura de um modelo SysML do satélite que contém os requisitos e parâmetros desejados para o sistema. A seguir O SatBudgets realiza diversas operações internas como: leitura e extração do modelo SysML, atribuição aos seus blocos correspondentes, aplicação das regras para o cálculo dos balanços mecânicos e elétricos, acesso ao SpaceESB para o cálculo dos balanços de termistores, comandos diretos e área do painel solar.

Toda troca de dados entre o SatBudgets e o SpaceESB é realizada através de arquivos XML envelopados pelo protocolo SOAP.

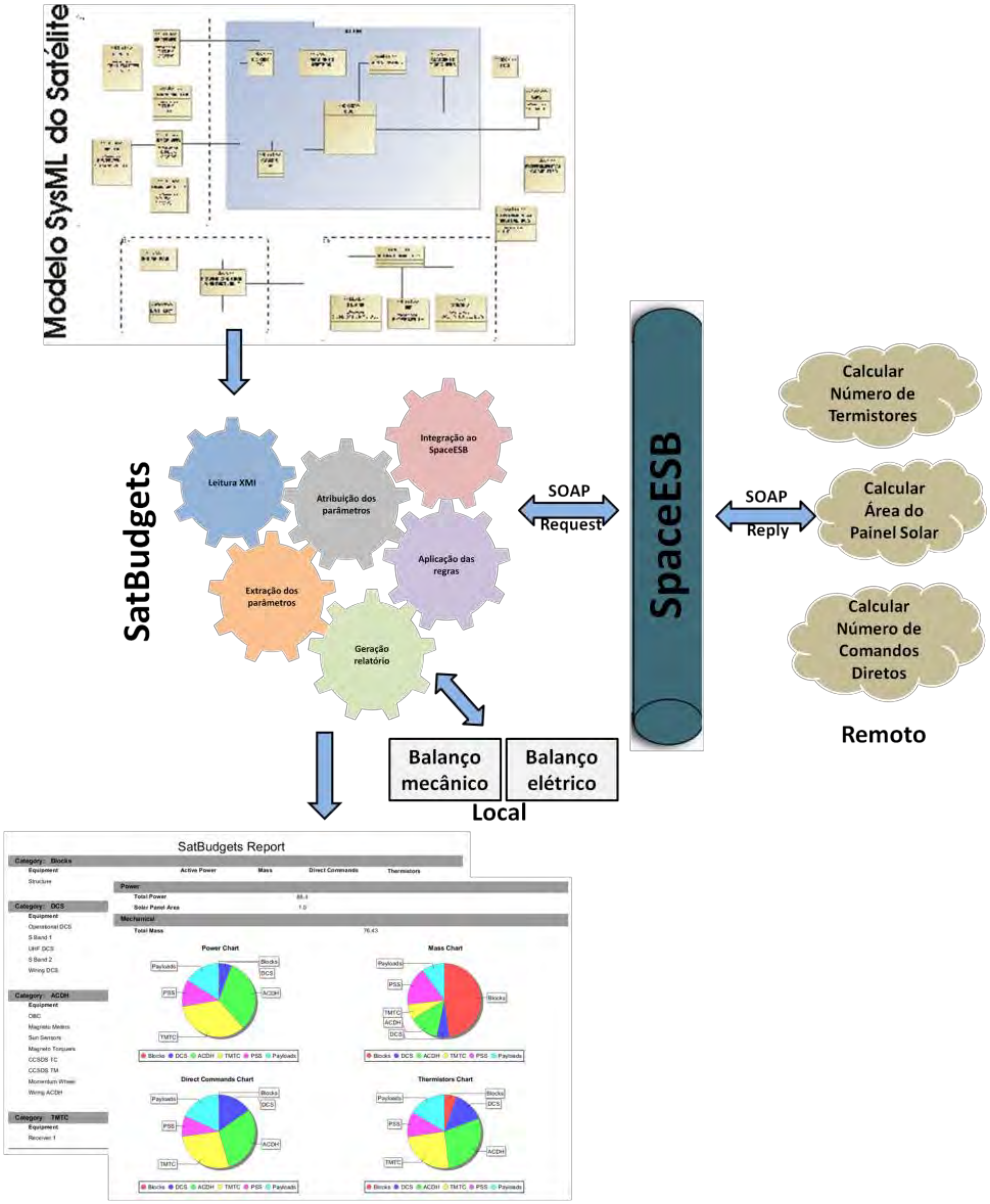



Figura 3.9 - Integração e execução entre SatBudgets e SpaceESB

A integração do barramento de serviços ao SatBudgets não exigiu grande esforço de desenvolvimento, sendo necessária apenas a criação de um método que fizesse o acesso ao SpaceESB. A Figura 3.10 mostra um trecho do código usado na integração. Nas linhas 1, 2 e 3 é feita a chamada à aplicação composta implantada no SpaceESB através de uma URL de acesso. Nas linhas de 4 a 7 são passados os parâmetros necessários para que os *web services* realizem os cálculos dos balanços.

```
1. org.netbeans.j2ee.wsdl.spaceesbbpel.inputdata.InputDataService service = new
                org.netbeans.j2ee.wsdl.spaceesbbpel.inputdata.InputDataService();
2. org.netbeans.j2ee.wsdl.spaceesbbpel.inputdata.InputDataPortType port =
            service.getInputDataPort();
3. ((BindingProvider)port).getRequestContext().put(BindingProvider.ENDPOINT_ADD
   RESS_PROPERTY,"http://localhost:9080/InputDataService/InputDataPort");

4. javax.xml.ws.Holder<java.lang.String> thermistorsReply = new
                javax.xml.ws.Holder<java.lang.String>();
5. javax.xml.ws.Holder<java.lang.String> directCommandsReply = new
                javax.xml.ws.Holder<java.lang.String>();
6. javax.xml.ws.Holder<java.lang.String> solarPanelReply = new
                javax.xml.ws.Holder<java.lang.String>();
7. port.inputDataOperation(serviceParameters, thermistorsReply,
   directCommandsReply, solarPanelReply);
```

Figura 3.10 - Trecho de código para acesso ao SpaceESB

Outra alteração realizada no SatBudgets foi para a apresentação do relatório dos balanços. O relatório gerado pelo SatBudgets apresenta o cálculo dos balanços e os gráficos desses cálculos. Porém, em sua versão original, o SatBudgets usa a ferramenta IReport para gerar o layout do relatório e toda a geração dos gráficos é realizada através de desenvolvimento de código. Já para a versão do SatBudgets integrada ao SpaceESB tanto o layout quanto os gráficos são gerados pelo IReport.

Para essa alteração foram necessárias, apenas, a criação de novas variáveis no IReport que recebessem os resultados obtidos com a execução dos *web services*, para que fossem exibidos pelo relatório. Com essa alteração, simples de ser realizada, eliminou-se linhas de código desnecessárias e que poderiam exigir futura manutenção.

Toda a arquitetura desenvolvida para o SpaceESB foi sendo testada passo a passo. Para isso, três níveis de integração foram implementados: integração em *localhost,* cliente-servidor e servidor-cliente-provedor. As subseções a seguir descrevem cada uma das evoluções na integração entre SatBudgets e SpaceESB. Este procedimento é um processo seguro e controlado de implementação e entrega de aplicações SOA.

### 3.4.1. Integração Localhost

A primeira etapa de integração realizada entre o SatBudgets e SpaceESB foi realizada em um ambiente local.

A integração em *localhost* representa um ambiente controlado. Neste ambiente tanto a ferramenta SatBudgets quanto o SpaceESB estão localizados na mesma máquina física, porém a comunicação estabelecida entre o SatBudgets e o SpaceESB é como uma comunicação entre máquinas remotas. Esse ambiente é útil para a finalidade de testes, assim como para usar recursos localizados na máquina atual, mas que se esperariam serem remotos. A Figura 3.11 mostra a integração em *localhost*.

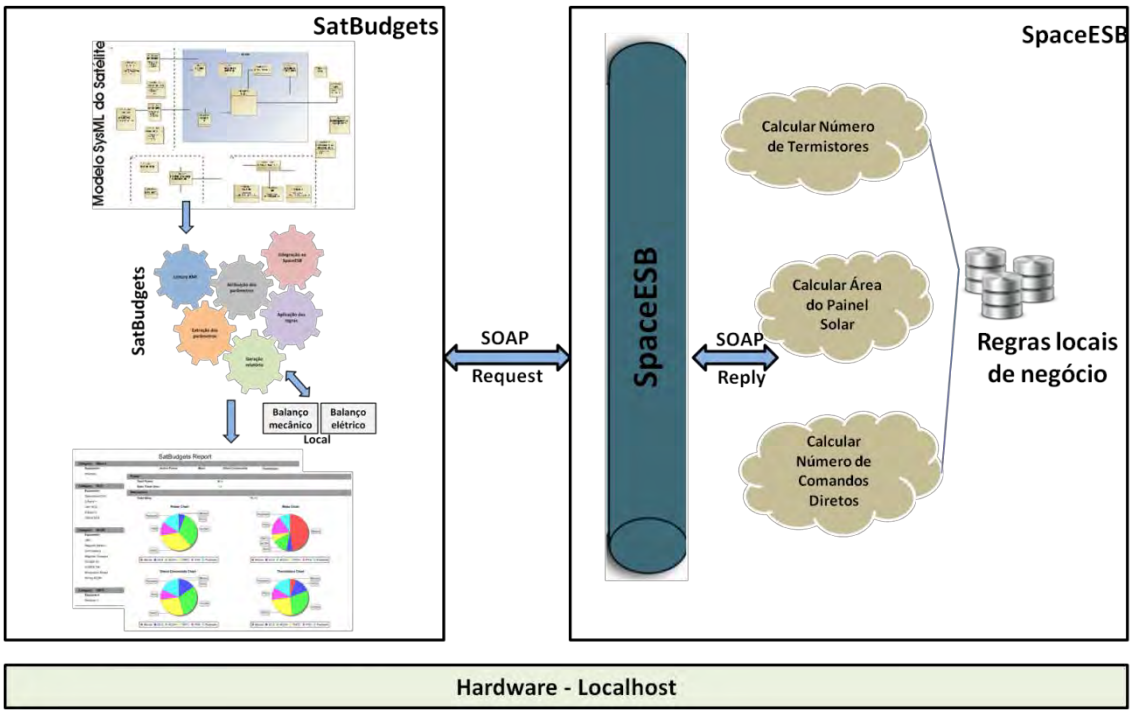



Figura 3.11 - Integração em ambiente *localhost*

### 3.4.2. Integração Cliente-Servidor

O segundo estágio de integração foi o cliente-servidor que é um modelo computacional que separa clientes e servidores, sendo interligados entre si geralmente utilizando-se uma rede local de computadores.

Para simular a integração entre cliente-servidor foram realizadas duas implementações. Na primeira, conforme a Figura 3.12, o servidor era uma máquina física e o cliente era executado em uma máquina virtual.

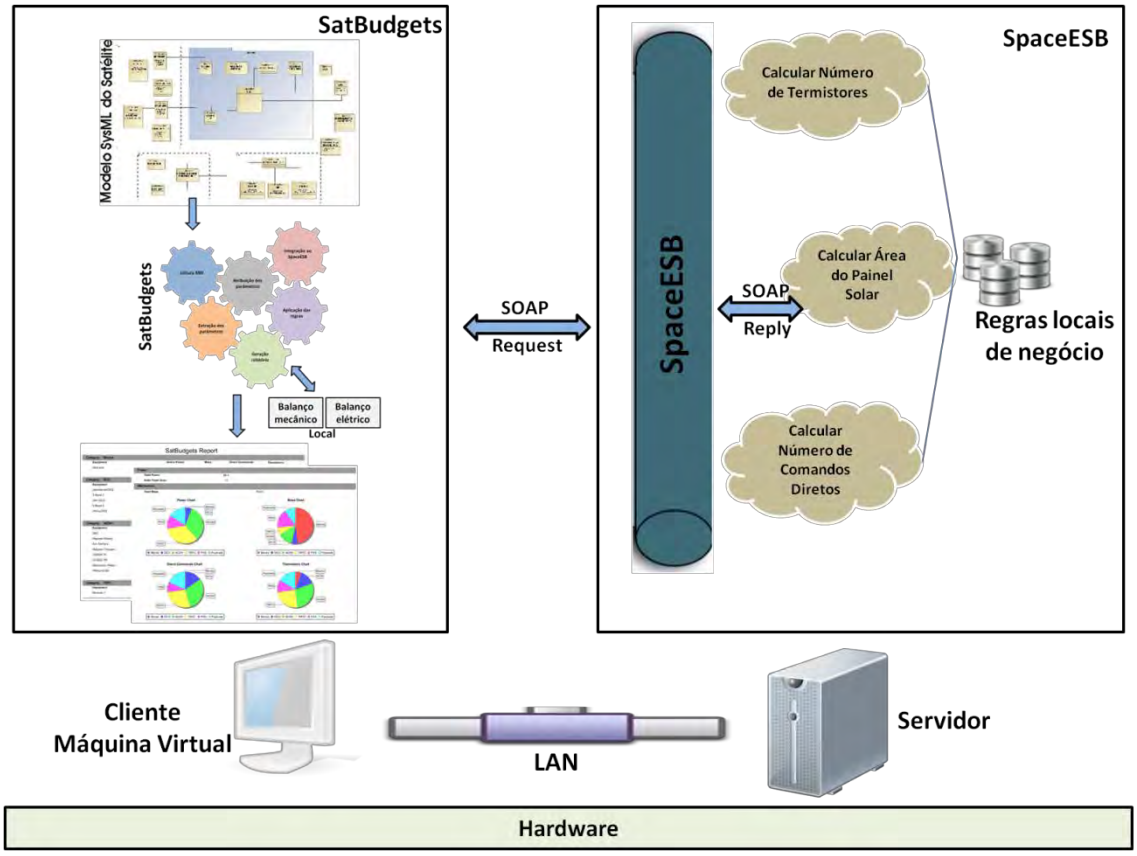



Figura 3.12 - Integração em ambiente cliente-servidor com máquina virtual

E na segunda implementação, tanto o cliente quanto o servidor eram diferentes máquinas físicas, conforme mostra na Figura 3.13.

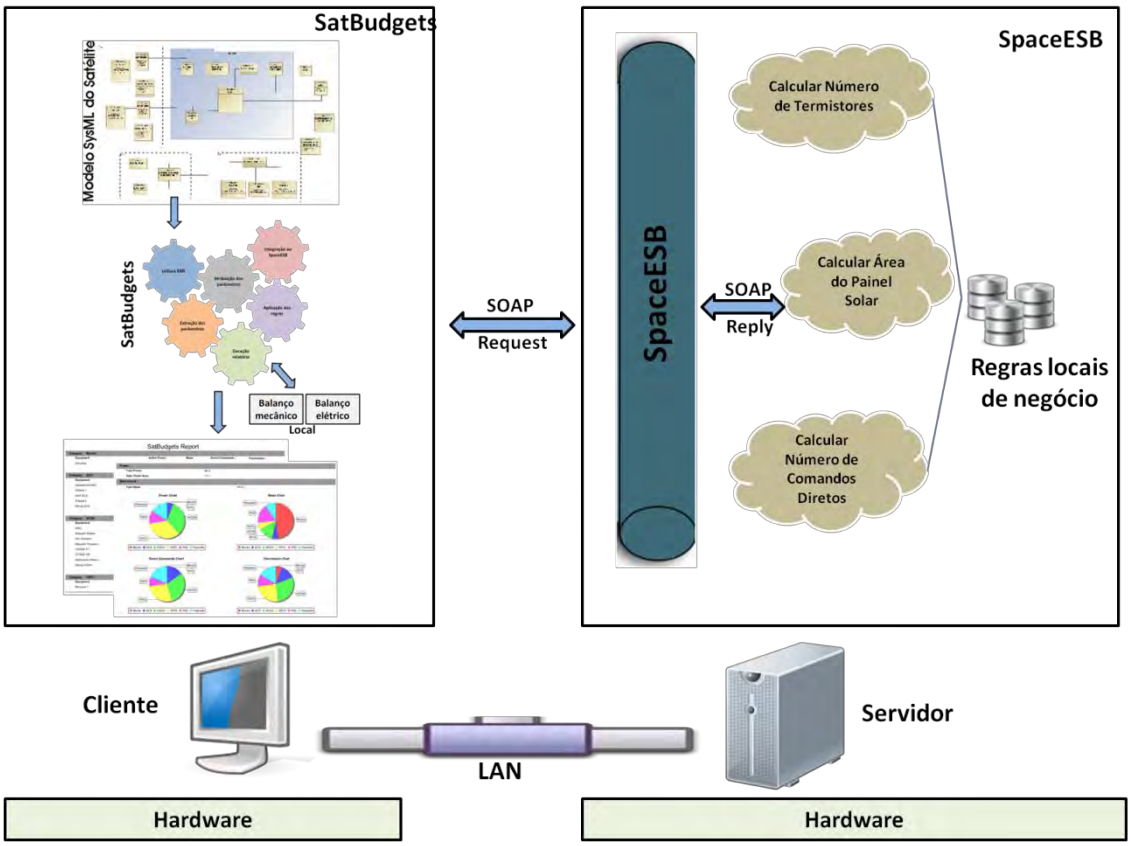



Figura 3.13 - Integração em ambiente cliente-servidor com diferentes máquinas físicas

A máquina virtual é o nome dado a uma máquina, implementada através de software, que executa programas como um computador físico. Para sua implementação a ferramenta utilizada foi o Microsoft Virtual PC (MICROSOFT, 2011).

Para a interligação em rede tanto no cenário cliente e servidor com diferentes máquinas físicas, quanto no cenário cliente e servidor com máquina virtual e máquina física foi utilizado um software chamado Hamachi, que simula uma rede local, ou *Local Area Network* (LAN) (LOGMEIN, 2011).

Para que o SatBudgets pudesse funcionar corretamente nos ambientes descritos foi necessário apenas mudar a URL de acesso no método que acessa o SpaceESB, conforme mostra a Figura 3.14.

```
1. org.netbeans.j2ee.wsdl.spaceesbbpel.inputdata.InputDataService service = new
                i. org.netbeans.j2ee.wsdl.spaceesbbpel.inputdata.InputDataService();
2. org.netbeans.j2ee.wsdl.spaceesbbpel.inputdata.InputDataPortType port =
                service.getInputDataPort();
3. ((BindingProvider)port).getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,"http://5.136.210.90:9080/InputDataService/InputDataPort");
```

Figura 3.14 - Alteração da URL para acesso ao SpaceESB

### 3.4.3. Integração Cliente-Servidor-Provedor

Nesta última etapa de integração, um ambiente totalmente distribuído foi preparado com o emprego de três diferentes máquinas: um nó para hospedar o SatBudgets, um nó para hospedar o ESB e um terceiro nó para hospedar os serviços que realizam os cálculos dos balanços, conforme mostrado na Figura 3.15.

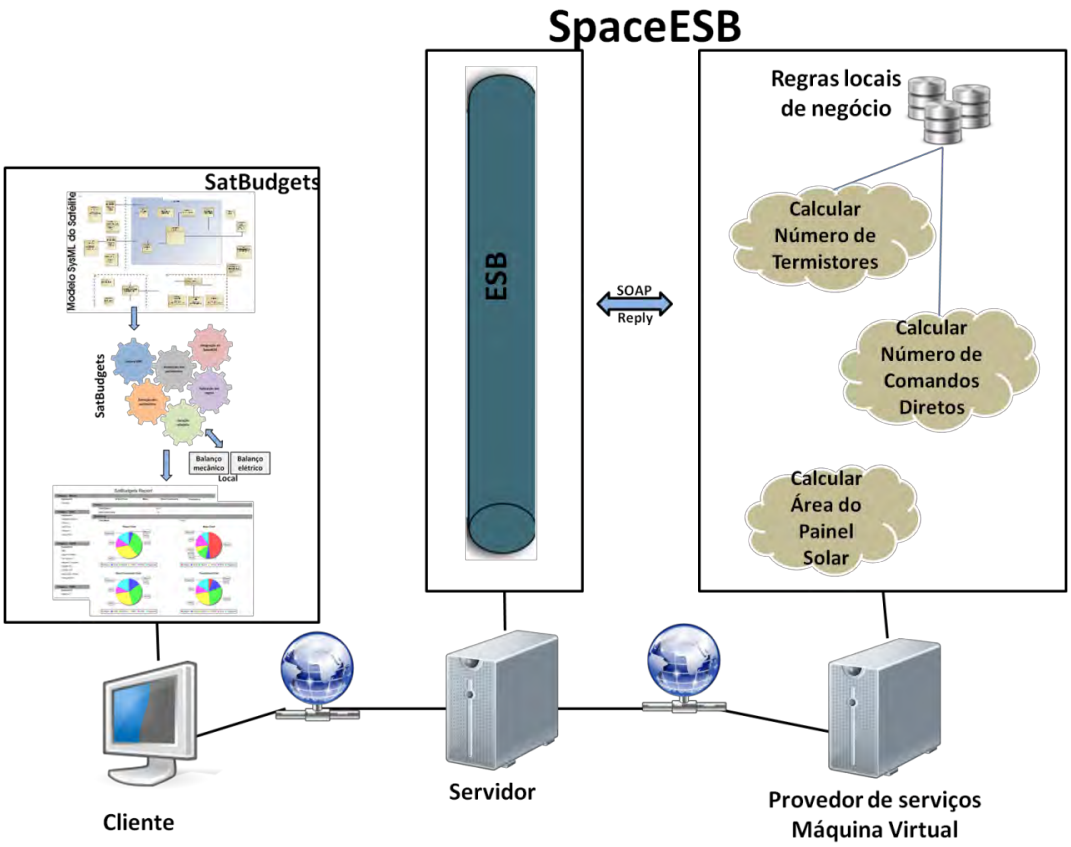



Figura 3.15 - Integração em ambiente cliente-servidor-provedor

Foram utilizadas duas máquinas físicas e uma máquina virtual. Nas máquinas físicas foram hospedados o cliente, com o SatBudgets, e o servidor, com o ESB. A máquina virtual foi utilizada como provedora dos serviços.

Assim como na integração cliente-servidor, esta integração utilizou o aplicativo Hamachi para simular a rede e atribuir diferentes IPs, tanto às máquinas físicas quanto às máquinas virtuais.

Como último passo para o correto funcionamento do ambiente distribuído, conforme mostrado nas Figuras 3.16 e 3.17, foi necessário alterar, na ferramenta SatBudgets, a URL de acesso ao ESB. Analogamente, no arquivo WSDL, deve-se alterar a URL de acesso aos serviços.

```
1. org.netbeans.j2ee.wsdl.spaceesbbpel.inputdata.InputDataService service = new
                    i. org.netbeans.j2ee.wsdl.spaceesbbpel.inputdata.InputDataService();
2. org.netbeans.j2ee.wsdl.spaceesbbpel.inputdata.InputDataPortType port =
            service.getInputDataPort();
3. ((BindingProvider)port).getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,"http://5.136.210.90:9080/InputDataService/InputDataPort");
```

Figura 3.16 - Alteração da URL para acesso ao ESB

```
<service name="InputDataService">
   <port name="InputDataPort" binding="tns:InputDataBinding">
      <soap:address location="http://5.147.221.166:8080/InputDataService"/>
   </port>
</service>
```

Figura 3.17 - Alteração da URL do arquivo WSDL para acesso aos serviços

## 3.5. Fluxo detalhado de operações

Ao iniciar o SatBudgets é exibida a tela principal da ferramenta, conforme apresentada na Figura 3.18.

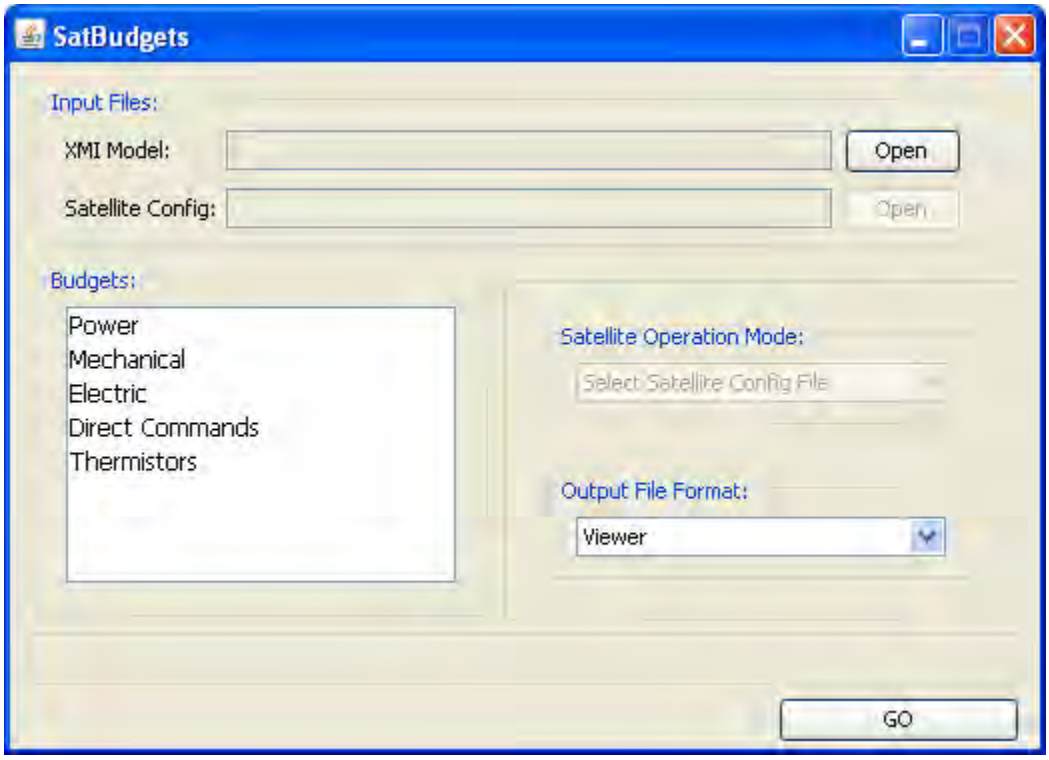



Figura 3.18 - Tela Principal do SatBudgets

O campo XMI *Model* é utilizado para se escolher qual o modelo SysML será utilizado no cálculo dos balanços.

O campo *Satellite Config* é utilizado para selecionar um diagrama de estados, onde estão descritos os modos de operação do satélite, entretanto esta funcionalidade foi desabilitada neste trabalho.

O campo *Budgets* apresenta a lista de balanços disponíveis na aplicação permitindo a seleção de qual, ou quais balanços serão calculados. O campo *Satellite Operation Mode* é usado para selecionar o modo de operação do satélite para o calculo do balanço, sendo os modos de operação carregados a partir do arquivo indicado no campo *Satellite Config*. Esta funcionalidade não foi utilizada por não fazer parte do escopo do trabalho.

Finalmente, o campo *Ouput File Format* é usado para selecionar o formato de saída do relatório, que pode ser escolhido entre *Viewer*, PDF, XLS e HTML. O botão “GO” é o responsável em dar início à execução da ferramenta.

Os campos *Satellite Config* e *Satellite Operation Mode* não foram implementados na versão original da ferramenta SatBudgets. Durante a integração do SatBudgets ao SpaceESB optou-se por usar a ferramenta o mais próximo possível de sua versão original.

Durante a fase de projeto conceitual é criado um modelo SysML contendo os requisitos do sistema pretendido. O diagrama SysML é um arquivo salvo no formato *XML Metadata Interchange* (XMI), sendo o modelo usado como parâmetro de entrada para o SatBudgets.

O SatBudgets realiza a leitura do modelo onde todos os parâmetros oriundos do modelo são recuperados, extraídos e atribuídos aos seus blocos correspondentes.

Os blocos do satélite e suas respectivas propriedades são, então, utilizados para o cálculo dos balanços. Para o cálculo dos balanços mecânicos e elétricos são aplicadas regras locais, já existentes na versão original da ferramenta SatBudgets, implementadas usando a API Drools.

Para o cálculo dos balanços no número de termistores, número de comandos diretos e área do painel solar, o SatBudgets se comunica com o SpaceESB e prove os parâmetros de entrada necessários para a execução dos balanços.

Assim sendo, o primeiro passo executado pelo SatBudgets é a criação, a partir do satélite, de um arquivo XML, denominado InputPar.xml, contendo: os blocos do satélite, a estratégia usada no controle térmico, a lista, número e consumo de energia dos equipamentos de cada bloco, e a área do painel solar. A Figura 3.19 mostra um trecho do arquivo XML gerado pelo SatBudgets.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <equipment>
        <category>DCS</category>
        <controlStrategy>passive</controlStrategy>
        <equipmentList>Operational DCS,S Band 1,UHF DCS,S Band 2,Wiring
        DCS</equipmentList>
        <number>1,1,1,1,1</number>
        <powerConsumption>5.0,5.0,0.0,0.0,0.0</powerConsumption>
        <solarPanelArea>1.0</solarPanelArea>
  </equipment>
  <equipment>
        <category>ACDH</category>
        <controlStrategy>passive</controlStrategy>
        <equipmentList>OBC,Magneto Meters,Sun Sensors,Magneto
         Torquers,CCSDS TC,CCSDS TM,Momentum Wheel,Wiring ACDH</equipmentList>
        <number>1,1,1,3,1,1,1,3</number>
        <powerConsumption>26.0,26.0,0.5,0.0,0.0,0.0,0.0,2.4
        </powerConsumption>
        <solarPanelArea>1.0</solarPanelArea>
  </equipment>
</configuration>
```

Figura 3.19 - Trecho do arquivo InputPar.xml gerado pelo SatBudgets

O arquivo XML gerado é usado como parâmetro de entrada para a execução dos serviços. A partir de então, o SatBudgets acessa o SpaceESB que por sua vez invoca os serviços.

Em seguida, os serviços extraem os parâmetros do arquivo XML e iniciam o cálculo do balanço requisitado. O cálculo do número de termistores e do número de comandos diretos utiliza arquivos de texto como fonte de regras que devem ser aplicadas para encontrar o número de termistores ou comandos diretos para cada equipamento.

A Figura 3.20 mostra um trecho dos arquivos de regras que encapsulam as lógicas de negócio de engenharia de sistemas mencionadas na seção 3. Foram criados três diferentes arquivos: passiveThermistorFactor.txt, activeThermistorFactor.txt, directCommandsFactor.txt. Nesses arquivos existe a lista de equipamentos que podem ser usados no projeto do sistema espacial

e o número (fator) recomendado, de termistores ou comandos diretos, para cada equipamento.

Com isso, no momento do cálculo do número de comandos diretos, por exemplo, o serviço acessa o arquivo directCommandsFactor.txt, encontra na lista o equipamento bateria, verifica qual é o fator correspondente ao equipamento e calcula o número de comandos diretos necessários multiplicando o fator pela quantidade de baterias existente do sistema (Equipamento=Batery, Fator=2, Quantidade de baterias=1, Fator*Quantidade de baterias = 2).

| Equipment-Factor | Equipment-Factor | Equipment-Factor |
|---|---|---|
| Structure-4 | Structure-4 | Structure-0 |
| Sun Sensors-0 | Sun Sensors-2 | Sun Sensors-0 |
| UHF-0 | UHF-2 | UHF-0 |
| Solar Panel-2 | Solar Panel-2 | Solar Panel-0 |
| OBC-2 | OBC-2 | OBC-2 |
| Diplex 1-0 | Diplex 1-2 | Diplex 1-0 |
| Momentum Wheel-1 | Momentum Wheel-2 | Momentum Wheel-0 |
| Battery-2 | Battery-2 | Battery-2 |

Figura 3.20 - Trechos dos arquivos com parâmetros de regra de negócio, passiveThermistorFactor.txt, activeThermistorFactor.txt e directCommandsFactor.txt, respectivamente.

Após a realização do cálculo dos balanços o SpaceESB retorna os resultados para o SatBudgets. A partir dos resultados de todos os balanços disponíveis no SatBudgets, a ferramenta pode realizar a última etapa de seu fluxo de trabalho, que é a geração do relatório contendo o balanço e sua apresentação aos responsáveis pelo sistema para que seu resultado seja avaliado. Se o resultado do balanço estiver conforme planejado, pode ser iniciada a fase de seleção de arquiteturas viáveis e posterior produção do sistema, caso contrário alterações no modelo podem ser feitas e novos balanços gerados até que o resultado desejado seja atingido. Nisto consiste o processo de “*System Level Trades*”

A Figura 3.21 apresenta um trecho do relatório gerado pela ferramenta SatBudgets original. Já a Figura 3.22 mostra o relatório após a integração ao SpaceESB, onde é possível visualizar os novos balanços disponibilizados pela aplicação. O relatório completo, tanto antes quanto após a integração, pode ser visto no Anexo B.

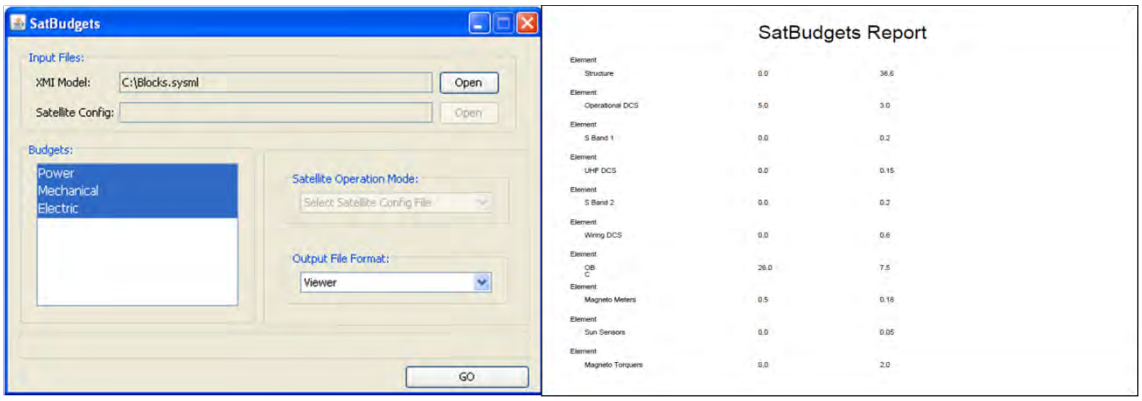



Figura 3.21 - Relatório do SatBudgets original

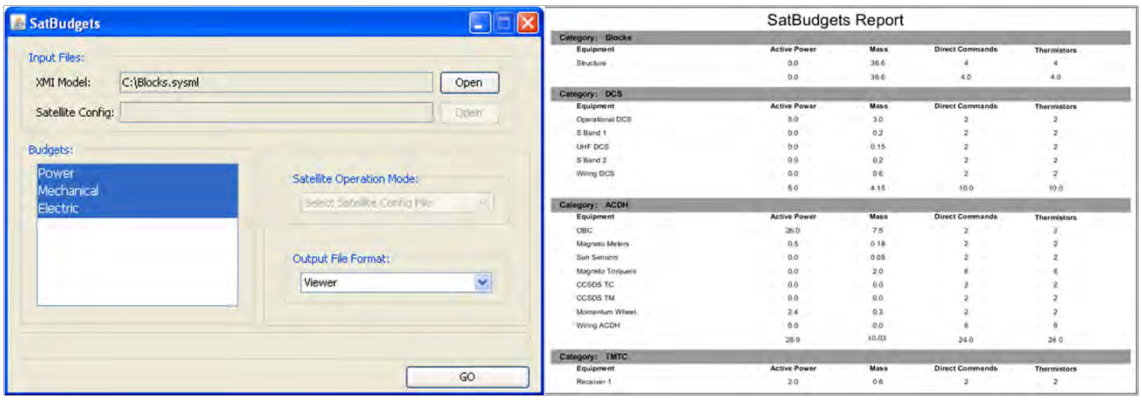



Figura 3.22 - Relatório do SatBudgets após a integração com SpaceESB

# 4 ESTUDO DE CASO E RESULTADOS

Neste capítulo será apresentado o estudo de caso realizado neste trabalho e os resultados obtidos.

Como já mencionado anteriormente, a ferramenta SatBudgets e o SpaceESB foram criados com a finalidade de apoiar e integrar processos da fase conceitual de projeto auxiliando os engenheiros envolvidos no projeto na tomada de decisão.

Na fase de projeto conceitual existe um processo chamado *System Level Trades*, conforme destacado na Figura 4.1 (LARSON; WERTZ, 1991), onde as possíveis arquiteturas (Arquitetura A e B) de um sistema espacial são combinadas e posteriormente selecionadas.

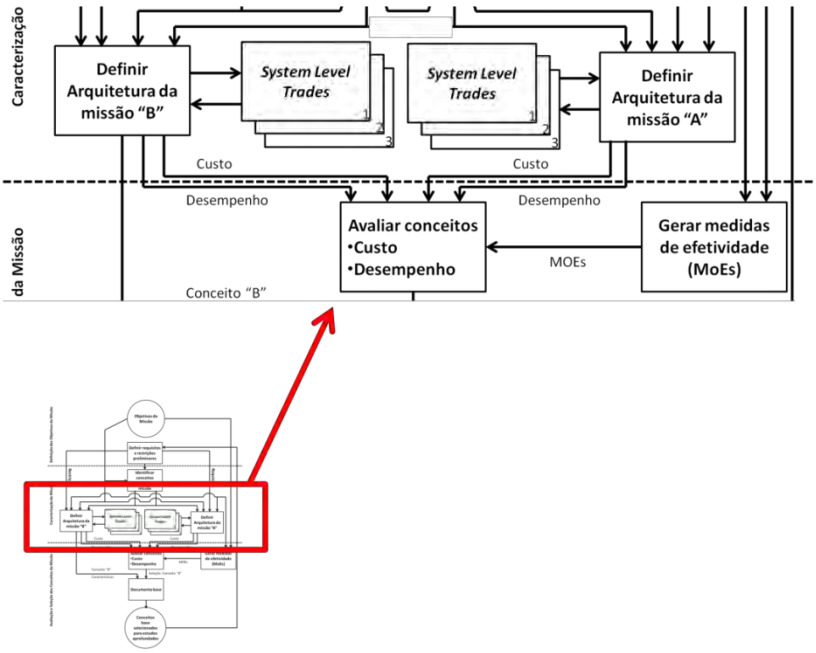



Figura 4.1 - Concepção e avaliação de arquiteturas viáveis para sistemas espaciais

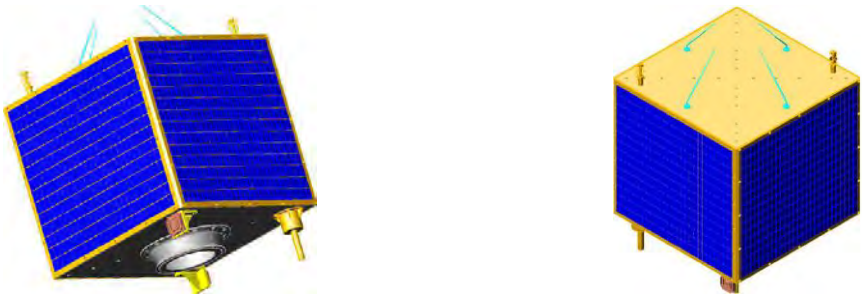
Neste capítulo é feita a exploração do espaço de projeto (*Design Space Exploration* - DSE) que é um aspecto central no projeto de sistemas. A DSE ajuda os engenheiros de projeto a escolher, entre as diversas combinações

possíveis alternativas, qual é a solução mais indicada para um determinado projeto (GANESAN; PREVOSTINI, 2006). Neste trabalho, o espaço explorado de arquiteturas foi propositalmente restrito a dois pontos por não ser o foco do trabalho.

Para isso foram criadas duas diferentes arquiteturas, que serão chamadas de: Arquitetura A (mais magra – Kg, menos gulosa – potência e menos confiável – sem redundância) e Arquitetura B (mais gorda – Kg, mais gulosa – potência e mais confiável – com redundância). Essas arquiteturas foram modeladas usando SysML e foram usadas como modelos de entrada para a ferramenta SatBudgets. Assim, a partir das saídas da ferramenta SatBudgets as arquiteturas criadas serão avaliadas nos seguintes balanços: mecânico, elétrico, de número de termistores, número de comandos diretos e área do painel solar. Esses compõem o repertório de balanços disponíveis atualmente.

## 4.1. Arquitetura A do satélite

Para arquitetura A o modelo SysML criado é baseado no satélite ITASAT, mostrado na Figura 4.2, um satélite universitário iniciado no ano de 2005, sendo desenvolvido por pesquisadores e alunos do INPE, Instituto Tecnológico de Aeronáutica (ITA) e de várias universidades brasileiras (ITASAT, 2005).



(a) Parte Inferior (b) Parte Superior

Figura 4.2 - Satélite ITASAT
Fonte: ITASAT (2005)

O ITASAT tem como missão formar especialistas na área de engenharia aeroespacial, desenvolver novas tecnologias para o setor e também participar do Sistema Nacional de Coleta de Dados. A Figura 4.3 representa o diagrama de blocos, criado em SysML para o ITASAT que servirá de base ao exercício de DSE a seguir com duas arquiteturas candidatas.

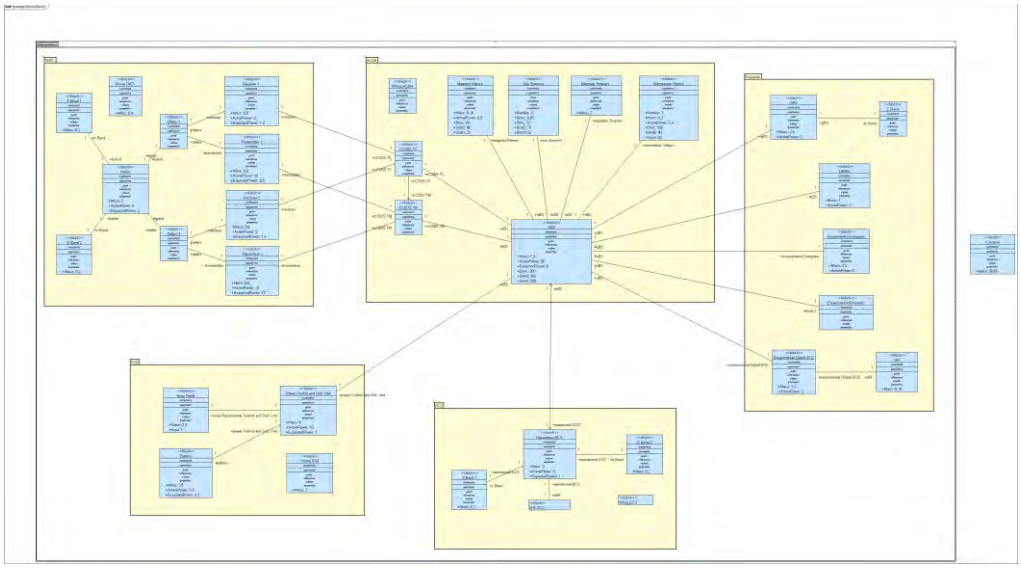


Figura 4.3 - Diagrama de Blocos base das arquiteturas A e B para o ITASAT

No diagrama de blocos são definidas suas características estruturais e comportamentais, bem como as relações hierárquicas, entre os blocos.

No modelo SysML de blocos são representados cinco pacotes que correspondem aos cinco subsistemas do satélite: *Telemetry and Telecommand* (TMTC), *Attitude Control and On-Board Data Handling* (ACDH), *Data Collection System* (DCS), *Power Supply Subsystem* (PSS) e *Payloads*. Cada um desses pacotes inclui vários blocos utilizados na arquitetura do satélite com as seguintes funções (SATO et al., 2011):

- A estrutura do satélite deve suportar os componentes do satélite e instrumentos da carga útil e distribuir as massas no interior do satélite a

fim de se obter os momentos de inércia adequados ao sistema de controle de atitude;

- O controle térmico controla e avalia a temperatura dos equipamentos embarcados, de modo passivo, garantindo sua operação dentro das margens de segurança;

- O PSS utiliza painéis solares montados sobre as faces laterais do satélite. Baterias armazenam energia para os períodos de eclipse e regulam a distribuição de energia do satélite;

- O ACDH usa a técnica de spin para estabilizar o satélite e contará com magnetômetro de três eixos, três sensores solares e três magneto torqueadores;

- O TMTC é composto por receptores e transmissores que operam em redundância. Os receptores operam em banda S;

- O computador de bordo é responsável por todo gerenciamento de bordo e é composto por processadores em redundância com alguns comandos diretos;

- O *Payload* inclui alguns experimentos como: o transponder digital de coleta de dados, o *MicroElectro-Mechanical-Systems* (MEMS), um *Global Positioning System* (GPS), experimento de comunicações e um experimento térmico;

- Os envelopes de massa, dimensão e potência são: massa menor que 80 Kg; dimensão = 700 x 600 x 400 mm, e potência de 100 W.

## 4.2. Arquitetura B do satélite

Para a arquitetura B do satélite o diagrama de blocos adotado foi o mesmo utilizado na Arquitetura A. Entretanto, foram realizadas apenas alterações nas

propriedades de cada bloco em termos de massa, energia consumida e quantidade de equipamentos. A Figura 4.4 mostra um comparativo entre as propriedades da arquitetura A e da arquitetura B para os blocos sensor solar e computador de bordo.

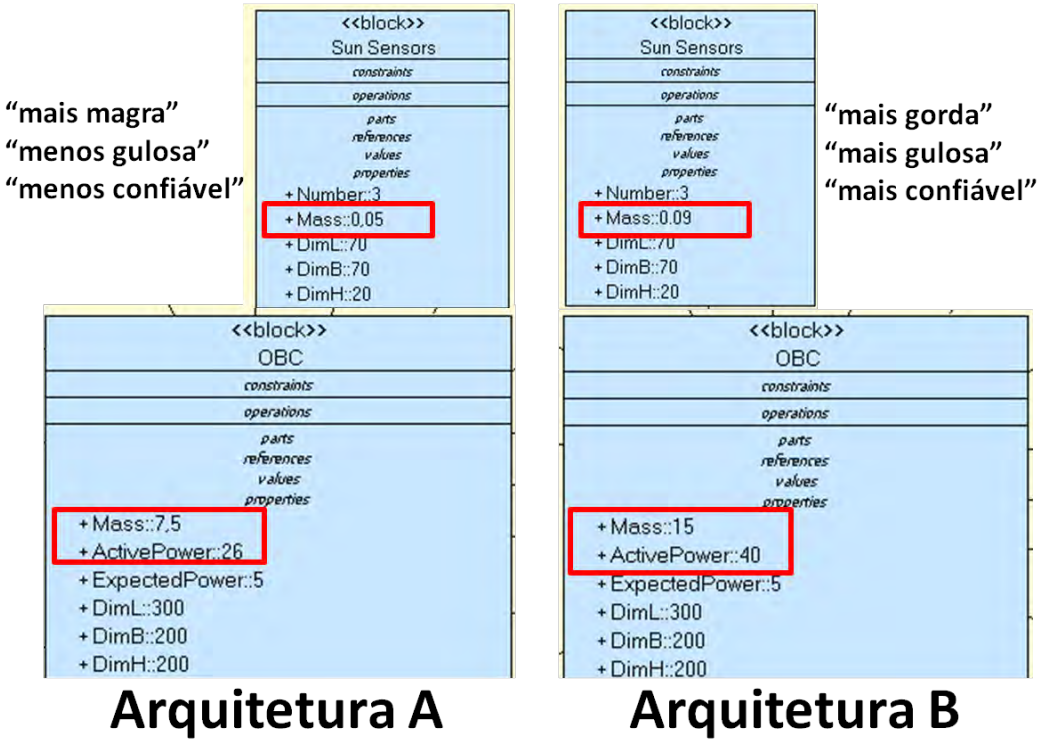



Figura 4.4 - Arquitetura A e arquitetura B (à direita) para os blocos sensor solar e computador de bordo

Conforme representado na Figura 4.4, para o computador de bordo (OBC) a massa teve seu valor duplicado, a quantidade foi alterada de um para dois e houve também uma alteração no consumo esperado de energia. Para os sensores solares houve apenas uma variação de massa. O objetivo dessas alterações é simular o uso de diferentes equipamentos em cada uma das arquiteturas.

Para os demais blocos da Arquitetura B o padrão de alteração das propriedades foi o mesmo empregado para os blocos, computador de bordo e sensor solar, sempre incrementando os valores definidos para a Arquitetura A.

## 4.3. Resultados de DSE entre Arquiteturas

Após a criação dos modelos SysML para a arquitetura A e para a arquitetura B a ferramenta SatBudgets foi executada tendo como parâmetro de entrada seus respectivos modelos.

A Figura 4.6(b) mostra um trecho do relatório gerado pelo SatBudgets para a arquitetura A e a Figura 4.6(c) representa um trecho do relatório para a arquitetura B. É possível observar, no relatório gerado, as diferenças nos números de comandos diretos, número de termistores, na massa e potência total, devido às diferenças existentes em cada arquitetura.

Os balanços calculados para a arquitetura A não apresentam problemas, a massa calculada foi de 76,43 Kg e a potência total é de 88,4 W o que mostra que a arquitetura está dentro do envelope de massa (80 Kg), e de potência (100 W). Já a arquitetura B apresenta problemas tanto para a massa quanto para a potência, pois ultrapassa os limites das especificações. A Figura 4.5 ressalta as principais diferenças entre ambas arquiteturas num gráfico Kiviat.

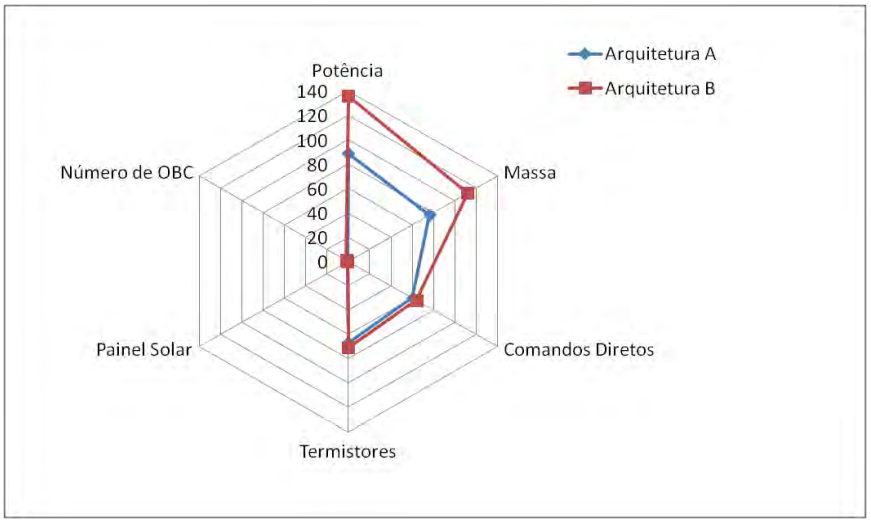
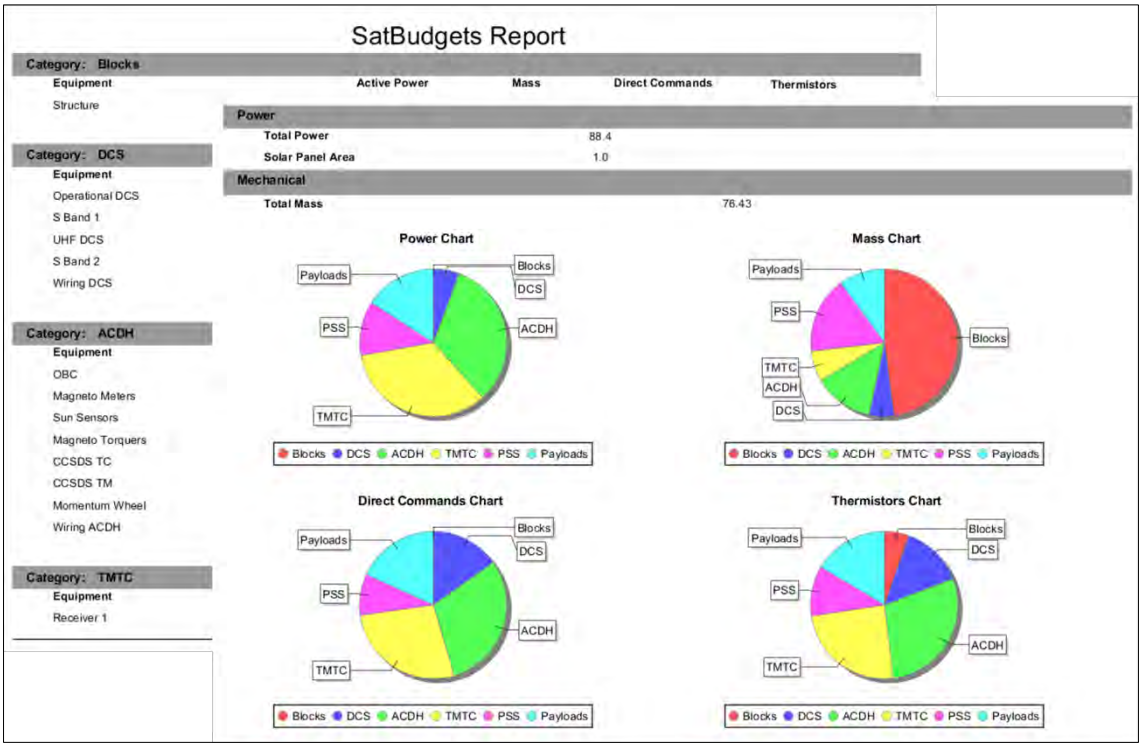


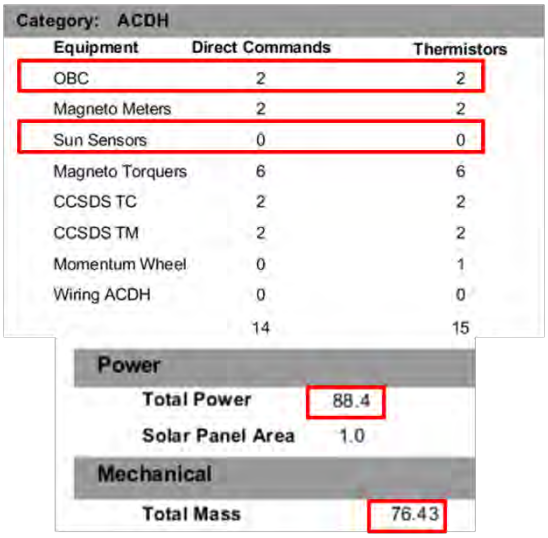

Figura 4.5 - Diferenças entre as arquiteturas A e B

Com base na análise dos relatórios gerados pelo SatBudgets os responsáveis pelo projeto irão avaliar qual arquitetura tem melhor custo-benefício para

satisfazer os objetivos da missão decidindo se o projeto deve, ou não, sofrer alguma alteração.



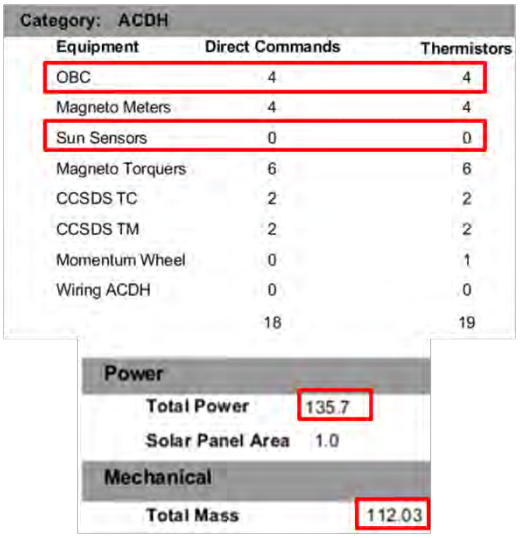

(a) Trecho do balanço gerado



| Category: ACDH | | |
|---|---|---|
| Equipment | Direct Commands | Thermistors |
| OBC | 2 | 2 |
| Magneto Meters | 2 | 2 |
| Sun Sensors | 0 | 0 |
| Magneto Torquers | 6 | 6 |
| CCSDS TC | 2 | 2 |
| CCSDS TM | 2 | 2 |
| Momentum Wheel | 0 | 1 |
| Wiring ACDH | 0 | 0 |
| | 14 | 15 |

| Power | |
|---|---|
| Total Power | 88.4 |
| Solar Panel Area | 1.0 |
| **Mechanical** | |
| Total Mass | 76.43 |

(b) Arquitetura A (sem problemas)



| Category: ACDH | | |
|---|---|---|
| Equipment | Direct Commands | Thermistors |
| OBC | 4 | 4 |
| Magneto Meters | 4 | 4 |
| Sun Sensors | 0 | 0 |
| Magneto Torquers | 6 | 6 |
| CCSDS TC | 2 | 2 |
| CCSDS TM | 2 | 2 |
| Momentum Wheel | 0 | 1 |
| Wiring ACDH | 0 | 0 |
| | 18 | 19 |

| Power | |
|---|---|
| Total Power | 135.7 |
| Solar Panel Area | 1.0 |
| **Mechanical** | |
| Total Mass | 112.03 |

(c) Arquitetura B (com problemas)

Figura 4.6 - Relatório gerado pelo SatBudgets

# 5 CONCLUSÕES

Neste capítulo serão apresentadas as considerações finais, as contribuições deste trabalho e sugestões de trabalhos futuros.

A perfeita concepção, construção e lançamento de sistemas espaciais, como um satélite, devido à natureza complexa e multidisciplinar, é bastante desafiadora. Especialistas de diversas áreas devem cooperar para garantir o sucesso do projeto. Algumas vezes, essas tarefas estão geograficamente e/ou temporalmente distribuídas exigindo um ambiente colaborativo e sempre acessível aos diversos parceiros de um projeto.

A fim de apoiar a fase conceitual de um projeto de satélite, este trabalho mostra como um conjunto simples de serviços pode ser distribuído e acoplado a um ambiente colaborativo dedicado ao domínio da engenharia de sistemas espaciais, chamado SpaceESB. Os serviços selecionados são relacionados às atividades de balanço iniciais como controle térmico, habilidade de comando e geração de energia, os quais afetam também outros subsistemas.

O mapeamento de uma atividade de fluxo de trabalho de projeto para um serviço em Java que implementa estas funcionalidades foi brevemente apresentado utilizando-se BPEL. Um arquivo WSDL foi gerado descrevendo a interface para o SpaceESB. A implementação foi feita usando apenas ferramentas de código aberto em todas as suas etapas.

Para demonstrar a facilidade de reuso de sistemas legados em uma Arquitetura Orientada a Serviços, foi realizada a integração da ferramenta SatBudgets ao SpaceESB. Através do consumo de serviços disponíveis no SpaceESB, a ferramenta SatBudgtes teve quantidade de balanços disponíveis aumentada.

A utilização de SOA juntamente com um ESB não é o único método possível para alcançar este resultado. Outra solução poderia incluir apenas a criação dos *web services* discutidos neste trabalho o que também promoveria reuso. Entretanto uma solução baseada em SOA oferece os seguintes benefícios:

- Manutenabilidade: a comunicação entre o consumidor e o fornecedor é baseada em interfaces bem definidas e padronizadas. Esta característica aumenta o poder de manutenabilidade dos sistemas, pois esconde os detalhes de implementação.
- Reusabilidade: com SOA é possível realizar a composição de novos serviços a partir de serviços já existentes, o que torna o desenvolvimento de aplicações mais rápido, aumenta qualidade e diminui custos e tempo de entrega.
- Flexibilidade: a necessidade de mudar um sistema para suportar novos requisitos é um fato dentro das organizações. Por isso, é necessário planejar arquiteturas que possam ser flexíveis e SOA traz diversas características que atendem a essa flexibilidade permitindo responder a as mudanças.
- Integração: sistemas legados e diferentes linguagens, plataformas e protocolos de comunicação, SOA pode facilitar a integração de sistemas heterogêneos já que a ideia é disponibilizar a lógica desses sistemas em forma de serviços interoperáveis.

O SpaceESB é apenas um começo para um ambiente de e-Engenharia de Sistemas Espaciais. Com todos os benefícios trazidos por uma Arquitetura Orientada a Serviços, é possível expandir sua abrangência adicionando mais processos da fase conceitual e até mesmo de outras fases do ciclo de vida do projeto de sistemas espaciais, bem como utilizar os sistemas já existentes apenas integrando-os ao barramento.

Para os diversos parceiros no projeto de um sistema espacial, a abordagem apresentada neste trabalho permite uma participação ativa no processo de tomada de decisão da concepção de um projeto espacial, maior agilidade, flexibilidade e a lidar com as pressões sobre os custos e o tempo, comuns neste domínio de sistemas.

## 5.1. Principais contribuições do trabalho

A seguir são listadas as principais contribuições obtidas no desenvolvimento deste trabalho:

- Melhoria do código do SatBudgets na geração de relatórios;
- Desenvolvimento de um processo de prototipação de *web services*, sendo três níveis de integração: *host*, cliente-servidor, cliente-servidor-provedor;
- Geração de ambiente colaborativo dedicado ao projeto conceitual de satélites, SpaceESB, a partir do uso de SOA e de um ESB;
- Integração do SatBudgets ao SpaceESB;
- Publicações técnicas (Anexo C).

## 5.2. Trabalhos Futuros

Este trabalho está inserido numa área de constante evolução e surgem novas normas e metodologias a cada dia. Apesar dos bons resultados obtidos, tanto na integração quanto no uso do SpaceESB, o ambiente pode ser dotado de novas funcionalidades de forma a ampliar as suas potencialidades, como:

- Ser internalizado ao trabalho da engenharia de sistemas do INPE e afins;
- Explorar as potencialidades do ESB em termos de segurança, e monitoramento dos serviços, etc.;
- Armazenar regras de negócio do sistema em uma base de dados;
- Integrar as bases de dados ao ambiente;
- Incrementar o número de serviços
- Mapeamento, modelagem de negócio e automação todos os processos da engenharia de sistemas.

## REFERÊNCIAS BIBLIOGRÁFICAS


AGUILAR, J. A.; DAWDY, A. B.; LAW, G. W. The aerospace corporations concept design center. In: ANNUAL INTERNATIONAL SYMPOSIUM OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING, 8., 1998, Vancouver, Canada. **Proceedings…** Disponível em: <http://www.tudelft.nl/live/binaries/8c2bdbe5-543b-4524-ba08-8e6cf12eb709/doc/paper03.pdf>. Acesso em: Maio de 2011.

ALMEIDA, F. J. Estudo e escolha de metodologia para o projeto conceitual. **Revista de Ciência e Tecnologia**. v. 8, n.16, p. 31–42, 2000.

BANDECCHI M., MELTON B.; ONGARO F. Concurrent Engineering Applied to Space Mission Assessment and Design. In: ESA BULLETIN, 99., 1999, Noordwijk, Holanda. **Proceedings...** Disponível em: <http://www.esa.int/esapub/bulletin/bullet99/bande99.pdf >. Acesso em: Agosto de 2010.

BERTRAND, R. **Conceptual design and flight simulation of space stations.** Munich, Germany: Herbert Utz Verlag, 1999. 128 p. ISBN 3896755005.

BPEL tutorial. **SearcSOA.com**. Disponível em: <http://searchsoa.techtarget.com/generic/0,295582,sid26_gci1330911_mem1,00.html?ShortReg=1&mboxConv=searchSOA_RegActivate_Submit&>. Acesso em: Junho de 2010.

CAVALCANTI, L.; ALMEIDA, A. Soa na Prática: Modelagem de Serviços com BPMN. **Revista Mundo Java.** n. 40, Ano VII, mar.2010.

CHAPPELL, D. A. **Enterprise service bus**. Sebastopol, CA.: O´Reilly Media, 2004.

DOS SANTOS, W. A. A Collaborative Framework to Support Space Systems Engineering. In: CHILEAN CONGRESS ON ELECTRICAL ENGINEERING, 12., Temuco, Chile, 1997. **Proceedings...** Temuco, 1997. p. 574-578.

DOS SANTOS, W. A.; LEONOR, B.B.F; STEPHANY, S. A Knowledge-Based and Model-Driven Requirements Engineering Approach to Conceptual Satellite Design. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, 28., 2009, Berlin. **Proceedings...** Heidelberg: Springer Verlag, 2009. v. 5829, p. 487-500.

ERL, T. **Service oriented architecture:** a field guide to integrating XML and web services. New Jersey, USA: The Prentice Hall, 2004. 560 p. ISBN 9780131428980.

_____. **SOA: Principles of Service Design**. USA: The Prentice Hall, 2007. 573 p. ISBN 9780132344821.

_____. **What is SOA:** An introduction to Service-Oriented Architecture. Disponível em: < http://www.whatissoa.com/>. Acesso em: Dezembro de 2011.

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION. **Space engineering –** Engineering design model data exchange (CDF)**.** Noordwijk, Netherlands: ECSS Secretariat, ESA-ESTEC Requirements & Standards Division, 20 Oct. 2010. (ECSS E-TM-10-25A).

_____. **Space project management:** organization and conduct of reviews. Noordwijk, Netherlands. ECSS Secretariat, ESA-ESTEC Requirements & Standards Division, 01 Sept. 1999. (ECSS-M-30-01A).

GANESAN, S.; PREVOSTINI, M. Bridging the Gap Between SysML and Design Space Exploration. In: FORUM ON SPECIFICATION AND DESIGN LANGUAGES. FDL 2006., 2006, Darmstadt, Germany. **Proceedings...** Darmstadt: Technische Universität Darmstadt, 2006. p. 389-395. ISBN: 9783000197109.

GOTTSCHALK, K.; GRAHAM, S.; KREGER, H.; SNEL, J. Introduction to Web services architecture. **IBM Systems Journal**, v.41, n.2, p. 170-177. 2002.

HARDWICK, A. **A Webservice-based collaborative systems engineering for space applications**. London: Birkbeck College, University of London, 2009. Submitted MSc Report to the School of Computer Science and Information Systems

HOLLEY, K.; ARSANJANI, A. **100 SOA questions asked and answered**. United States of America: Prentice Hall, 2010. 304 p. ISBN 9780137080205.

INSTITUTO TECNOLÓGICO AEROESPACIAL (ITA). **ITASAT -** satélite universitário. São José dos campos , 2005. Disponível em: <www.itasat.ita.br>. Acesso em: out. 2011.

JOSUTTIS, N. M. **SOA in Practice: The Art of Distributed System Design**. United States of America: O'Reilly, 2007. 352 p. ISBN 9780596529550.

KEEN, M. et al. **Patterns:** implementing an SOA using an enterprise service bus. United States of America: IBM Redbooks, 2004. 380 p. ISBN 9780738490007.

LARSON, W. J.; WERTZ, J. R. **Space mission analysis and design**. 2. ed. Dordrecht, Netherlands: Microcosm Press, 1991. 884 p. ISBN 9781881883012.

LAZZERI, J. C. **SOA -** modelando serviços com fluxos em BPMN. 2010. Disponível em: < http://www.artigonal.com/ti-artigos/soa-modelando-servicos-com-fluxos-em-bpmn-3745520.html>. Acesso em: Ago. 2011.

LEONOR, B. B. F. **Um enfoque baseado em conhecimento e dirigido a modelos de engenharia de requisitos para projeto conceitual de satélites**. 2010. 111 p. (sid.inpe.br/mtc-m19@80/2010/01.27.12.06-TDI). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2010. Disponível em: <http://urlib.net/8JMKD3MGP7W/36RJ89B>. Acesso em: jan. 2011.

LEWIS,B. ; LANG, J.; JOLLY, R. Modular concurrent engineering models: enabling alternative models in conceptual satellite design**.** In: AEROSPACE CONFERENCE, 2007 , Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2007. p. 1-8. ISBN 1424405254.

LOGMEIN. **LogMeIn Hamachi**. 2011. Disponível em: <https://secure.logmein.com/BR/products/hamachi/features.aspx>. Acesso em: set. 2011.

LONGO J. S. C. **SOA with OpenESB**, sun microsystems. 2009. Disponível em: <http://api.ning.com/files/Sn6i83r-auY8oBXNNYPn61ZsEMqGd3sU-421*dPFgL0a9aX*MnF0rHU0uZzcxDGOxOq8WCIDX5pk6XchShvtTKV*YL3lfEc/SOAcomOpenESB.pdf>. Acesso em: Outubro de 2010.

MEIJERS, M. A. H. A.; HAMANN, R. J.; ZANDBERGEN, B. T. C. Applying systems engineering to university satellite projects. In: ANNUAL INTERNATIONAL SYSMPOSIUM INCOSE, 14., 2004, Toulouse. **Proceedings...** Toulouse: INCOSE, 2004. p. 11–19.

MICROSOFT. **Virtual PC 2007**. Disponível em: <http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=4580#overview>. Acesso em: ago. 2011.

MOREIRA, J. J. M**. wsQL –** uma arquitectura de software baseada em Web services.2005. 208 p. Dissertação (Mestrado em Tecnologia Multimídia) - Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2005. Disponível em: <http://repositorio- aberto.up.pt/bitstream/10216/11646/2/Text o integral.pdf>. Acesso em: jul. 2011.

MORO, T. D., DORNELES, C. F., REBONATTO, M. T. **Web services WS-* versus Web Services REST**. 2009. Disponível em: <http://www.upf.br/computacao/images/stories/TCs/arquivos_20092/Tharcis_Dal_Moro.pdf>. Acesso em: Novembro de 2010.

MORSE, E. et al. Next generation concurrent engineering: developing models to complement point designs. In: IEEE AEROSPACE CONFERENCE, 14., Big Sky, MT, USA. **Proceedings...** Big Sky: IEEE, 2006. p. 1-15. ISBN 078039545X.

NETBEANS. **NetBeans IDE**. Disponível em: <http://netbeans.org/community/releases/67/>. Acesso em: Julho de 2010.

OASIS STANDAR. **Reference model for service oriented architecture 1.0**. 2006. Disponível em: <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf>. Acesso em: ago.2011.

_____. **UDDI Spec TC**. UDDI Version 3.0.2, UDDI Spec Technical Committee Draft, Dated 20041019. Disponível em: <http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm#_Toc85907967>. Acesso em: ago.2011.

OPENESB. **OpenESB introduction**. Disponível em: <http://wiki.open-esb.java.net/Wiki.jsp?page=OpenESBIntroductionScreencast>. Acesso em: set. 2011.

OPEN GROUP. **SOA reference architecture** - Draft Technical Standard April 2009. Disponível em: < http://www.opengroup.org/projects/soa-ref-arch/uploads/40/19713/soa-ra-public-050609.pdf>. Acesso em: jul. 2011.

RODRIGUEZ, K.; ALASHAAB, A. Knowledge web-based system architecture for collaborative product development. **Computers in Industry Journal** v. 56, p. 125-140, 2005.

ROSS, J. et al. **Enterprise architecture as strategy**: creating a foundation for business execution. Cambridge: Harvard Business School Press, 2006.

SATO, L. H. S.; YAMAGUTI, W.; FERNANDES, D. ITASAT-1: uma proposta de continuidade do Sistema Brasileiro de Coleta de Dados Ambientais. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 15. (SBSR), 2011, Curitiba. **Anais...** São José dos Campos: INPE, 2011. p. 9017-9023.ISBN 9788517000577.

SHIN S. **SOA using OpenESB, BPEL, and NetBeans**. 2007. Sun Microsystems. Disponível em: <http://docs.huihoo.com/openesb/soabpelopenesb.pdf>. Acesso em: Nov. 2010.

SMITH, P. L. et al. Concurrent Design at Aerospace. **Crosslink: The Aerospace Corporation magazine of advances in aerospace technology**. v. 2, n. 1, 2001. Disponível em: <http://www.aero.org/publications/crosslink/winter2001/01.html>. Acesso em: Agosto de 2011.

SOMEKH M. **Introduction to Sun GlassFish Enterprise Service Bus**. 2008. Software Infrastructure Group Sun Microsystems. Screencast. Disponível em: <http://developers.sun.com/docs/javacaps/tutorials/screencasts/Composite/start.html>. Acesso em: jan. 2011.

SOMMERVILLE, I. **Engenharia de Software**. 8 ed., Brasil: Addison-Wesley, 2007. p. 568. ISBN 9788588639287.

SOUZA, P.N. **Curso introdutório em tecnologia de satélites**. São José dos Campos: ETE - INPE, 2008.

W3C. **Simple Object Access Protocol (SOAP) 1.1.** 2000: W3C Note 08 May 2000. Disponível em: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>. Acesso em: Outubro de 2010.

_____. **Web Services Architecture**. 2002. W3C Working Draft 14 November 2002. Relatório Técnico. Disponível em: <http://www.w3.org/TR/2002/WD-ws-arch-20021114/>. Acesso em: jul. 2011.

_____. **Web Services Architecture**. 2004. W3C Working Group Note 11 February 2004. Relatório Técnico. Disponível em: < http://www.w3.org/TR/ws-arch/>. Acesso em: jul. 2011.

_____. **Web Services Description Language (WSDL) 1.1**. 2001. W3C Note 15 March 2001. Disponível em: <http://www.w3.org/TR/wsdl>. Acesso em: ago. 2011.

WALL, S. D. Use of Concurrent Engineering in Space Mission Design**.** In: EUROPEAN SYSTEMS ENGINEERING CONFERENCE, EuSEC 2000., 2000, Munich, Germany. **Proceedings...** Munich, 2000.

WATKINS, R.; MCARDLE, M.; LEONARD, T.; SURRIDGE, M. Crossmiddleware Interoperability in Distributed Concurrent Engineering. In: INTERNATIONAL GRID INTEROPERABILITY AND INTEROPERATION WORKSHOP, IGIIW 2007., 2007, Bangalore, India. **Proceedings...** Disponível em: <http://eprints.ecs.soton.ac.uk/15116/1/watkinsCrossInteropDistConEngIEEE.pdf> Acesso em: Abril de 2011.

WILKE, M.; QUIRMBACH, O.; SCHIFFNER, M.; IGENBERGS, E. MuSSat – A Tool for Satellite Design in a Concept Design Center, In: EUROPEAN SYSTEMS ENGINEERING CONFERENCE, EuSEC 2000., 2000, Munich, Germany. **Proceedings...** Munich, 2000.

WU Q.; ZHOU C., JING T. Research on SOA based framework of collaborative design system for complicated product. In: INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND SOFTWARE ENGINEERING, 1., 2008. Wuhan, China. **Proceedings...** Wuhan, 2008.

## ANEXO A - MELHORIAS REALIZADAS NOS WEB SERVICES DE HARDWICK, (2009)

As regras de negócio aplicadas por Hardwick, (2009) para o desenvolvimento dos web services, foram simples. A linguagem de programação em que os serviços foram desenvolvidos foi Java.

Neste trabalho a simplicidade e a linguagem de programação foram mantidas para o desenvolvimento. Porém duas mudanças principais foram realizadas para melhorar o código de Hardwick, (2009).

A primeira mudança foi com relação as regras de negócio, em Hardwick, (2009) as regras estavam hard-coded, para determinar, por exemplo, o número de termistores que um determinado equipamento deveria receber, o código fazia a verificação através de comparação de strings, conforme mostrado na Figura A.1.

```
if(equipmentItemType.compareTo("sun sensor") == 0){
                    thermistorFactor = 0;
                }
                else if((equipmentItemType.compareTo("obc")
== 0)||(equipmentItemType.compareTo("solar array") == 0) ||
(equipmentItemType.compareTo("battery") == 0)){
                    thermistorFactor = 2;
                }
```

Figura A.1 - Comparação de *strings hard-coded*

Na nova versão do web service, usada no SpaceESB, as regras de negócio foram movidas para um arquivo de dados que é consultado e retorna o valor de termistores que um determinado equipamento deve receber.

A segunda grande alteração foi a mudança no tipo dos parâmetro de entrada dos web services. Em Hardwick, (2009) os parâmetros de entrada eram strings que continham a categoria de cada equipamento, a lista de equipamentos, a quantidade de equipamentos e o consumo de energia de cada equipamento, Conforme mostrado na Figura A.2.

```
String equipmentCategoryList = "Power#Power#Power#ACDH#ACDH#ACDH#ACDH#TM /
TC#TM / TC#TM / TC#TM / TC#TM / TC#TM / TC#TM / TC#TM /
TC#Payloads#Payloads#Payloads#Payloads#Payloads#Payloads#Miscellaneous#Miscella
neous#Miscellaneous";

String equipmentTypeList = "PCDU#Solar Array#Battery#Magneto Torquers#3-Axis
Magnetometer#Sun Sensor#OBC#Transceiver#Diplex#Hybrid#S-Band Antenna#Rx UHF
DCS Antenna #Tx S Band DCS Antenna#Digital UHF DCS Antenna#GPS L
Antenna#GPS#Operational DCS#Radiation Effect Exp#MEMS#Experimental Digital
DCS#Experimental OBC#Thermal Control #Harness#Structure";

String equipmentCountList = "1#1#1#3#1#3#1#2#2#1#2#1#2#1#1#1#1#1#1#1#1#1#1#1#1";

String equipmentPowerConsumptionList =
"0#0#0#1#2.4#0#26#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0";
```

Figura A.2 - Parâmetros de entrada em *strings*

Na versão melhorada, conforme mostrado na Figura A.3, os web services recebem como parâmetro de entrada um arquivo XML com uma estrutura de árvore XML, organizando melhor os parâmetros de entrada.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <equipment>
        <category>DCS</category>
        <controlStrategy>passive</controlStrategy>
        <equipmentList>Operational DCS,S Band 1,UHF DCS,S Band 2,Wiring
        DCS</equipmentList>
        <number>1,1,1,1,1</number>
        <powerConsumption>5.0,5.0,0.0,0.0,0.0</powerConsumption>
        <solarPanelArea>1.0</solarPanelArea>
  </equipment>
  <equipment>
        <category>ACDH</category>
        <controlStrategy>passive</controlStrategy>
        <equipmentList>OBC,Magneto Meters,Sun Sensors,Magneto
         Torquers,CCSDS TC,CCSDS TM,Momentum Wheel,Wiring ACDH</equipmentList>
        <number>1,1,1,3,1,1,1,3</number>
        <powerConsumption>26.0,26.0,0.5,0.0,0.0,0.0,0.0,2.4
        </powerConsumption>
        <solarPanelArea>1.0</solarPanelArea>
  </equipment>
</configuration>
```

Figura A.3 - Parâmetros de entrada em XML

## ANEXO B – TELA PRINCIPAL E RELATÓRIO DO SATBUDGETS

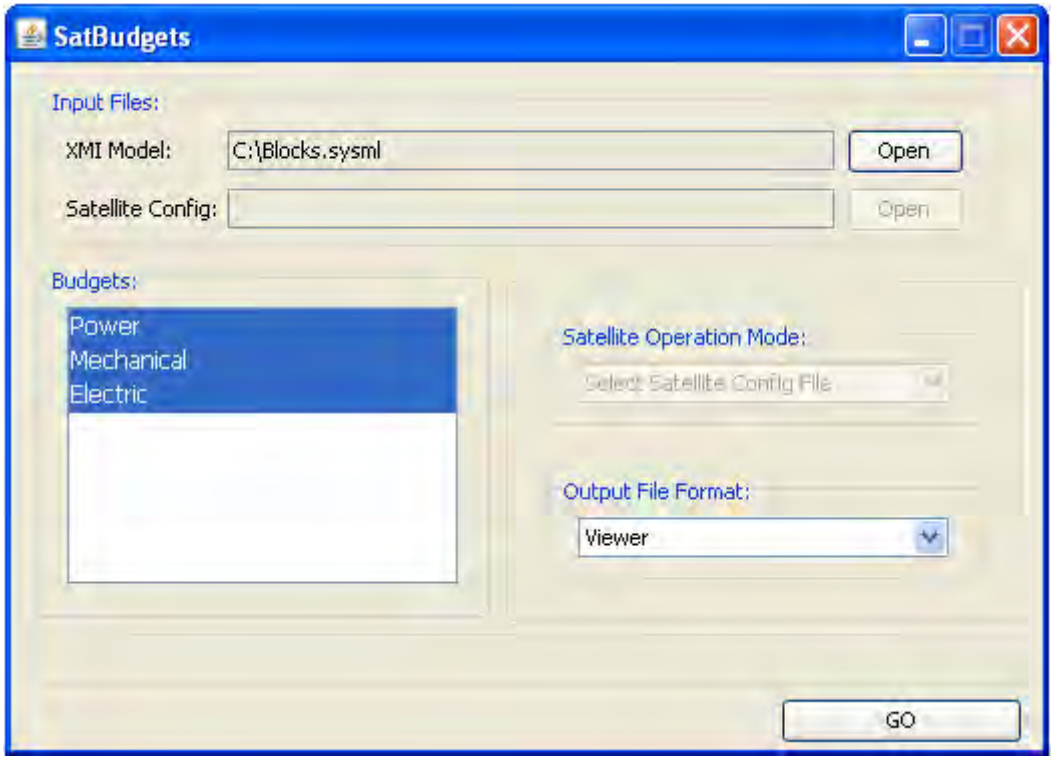
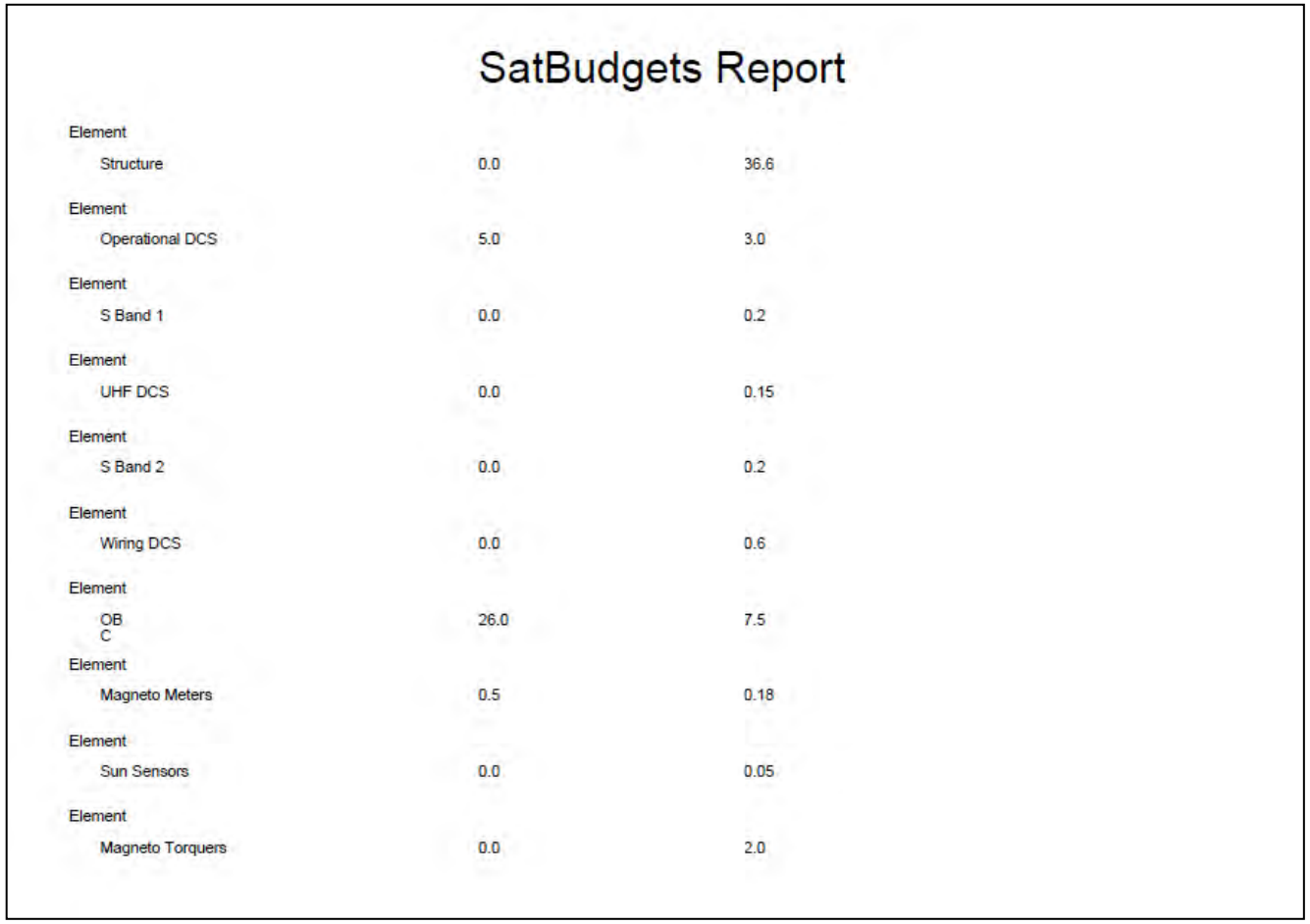


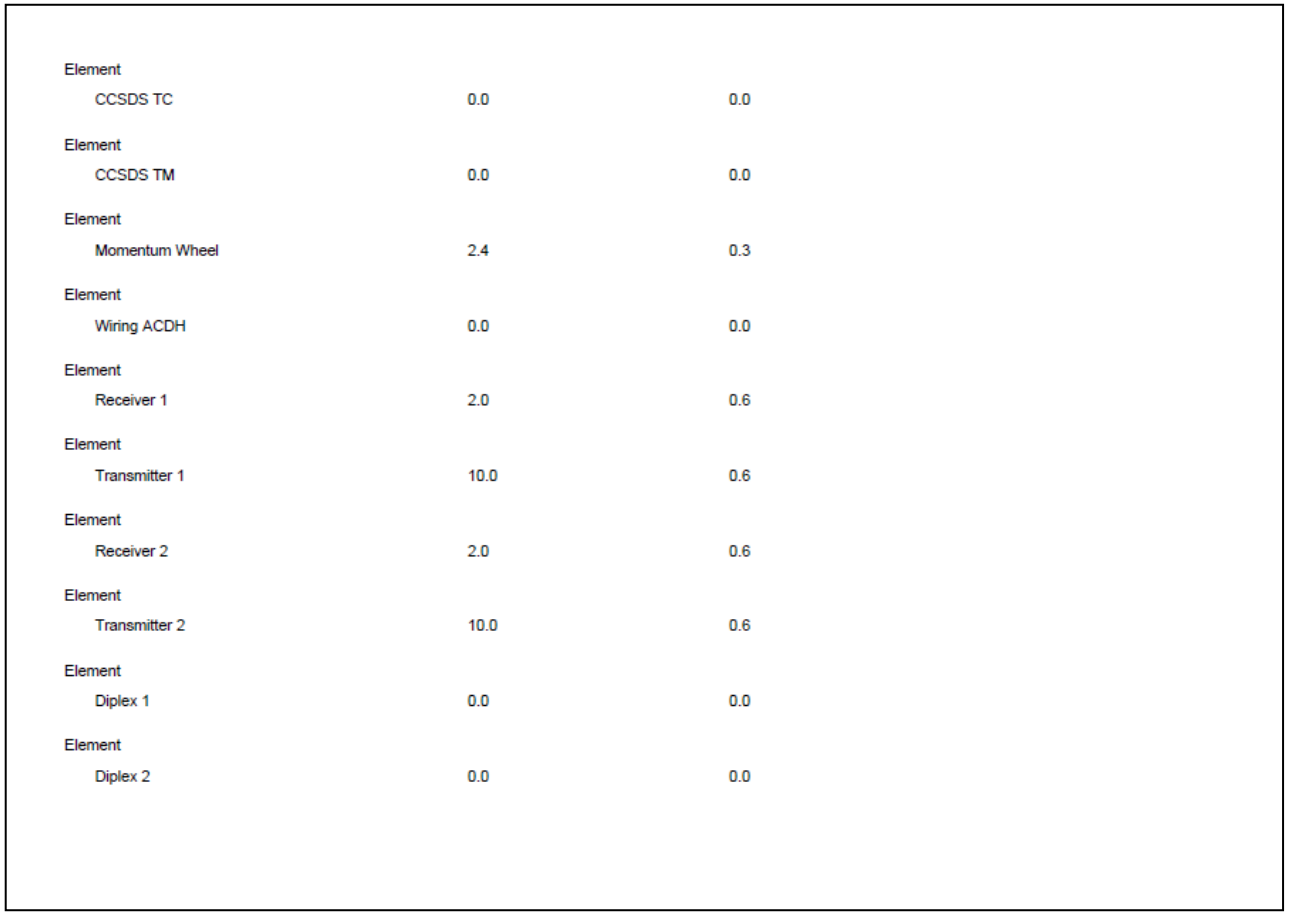

Figura B.1 - Tela principal do SatBudgets antes da integração com SpaceESB



SatBudgets Report

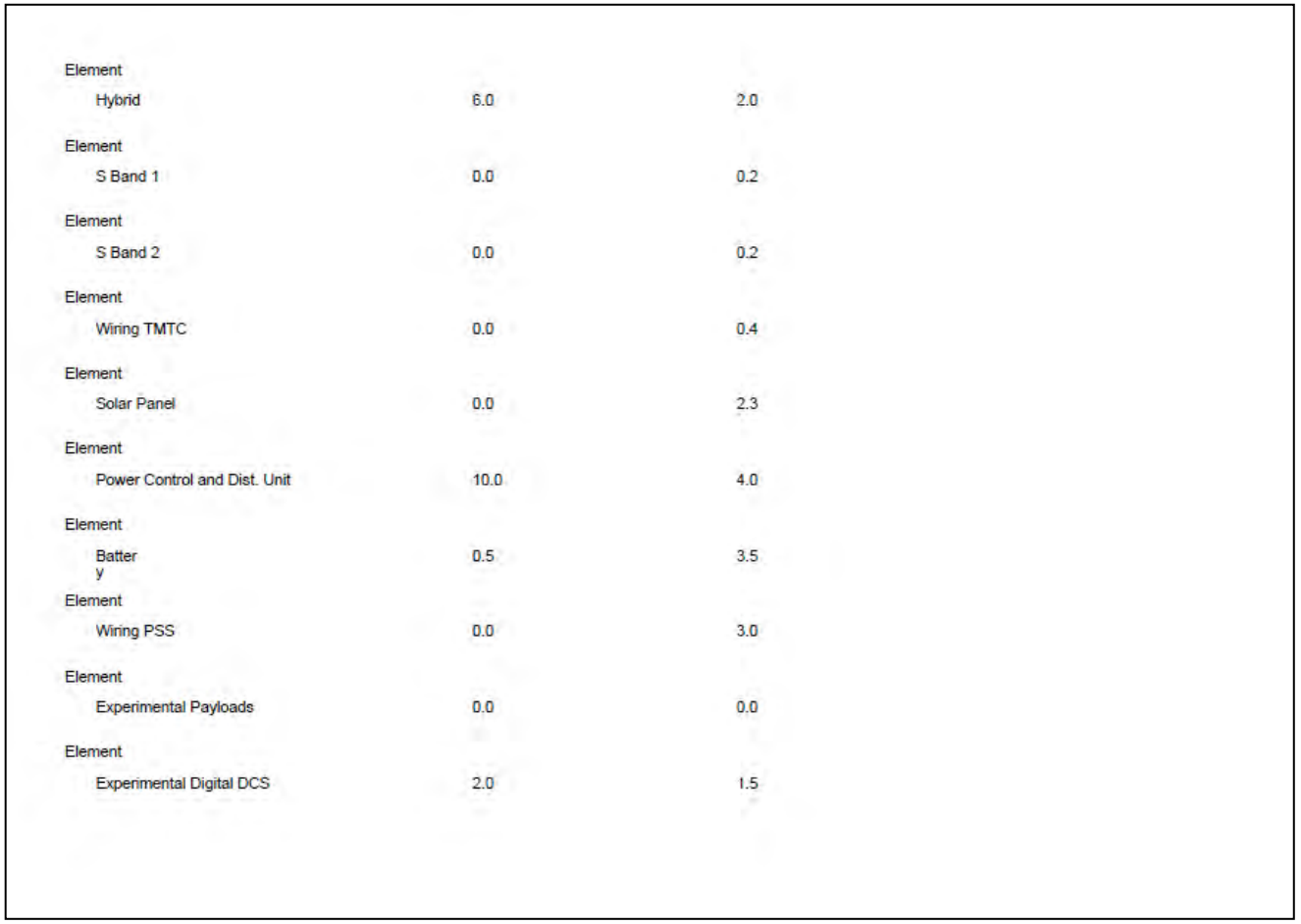
| | | |
|---|---|---|
| Element | | |
| Structure | 0.0 | 36.6 |
| Element | | |
| Operational DCS | 5.0 | 3.0 |
| Element | | |
| S Band 1 | 0.0 | 0.2 |
| Element | | |
| UHF DCS | 0.0 | 0.15 |
| Element | | |
| S Band 2 | 0.0 | 0.2 |
| Element | | |
| Wiring DCS | 0.0 | 0.6 |
| Element | | |
| OBC | 26.0 | 7.5 |
| Element | | |
| Magneto Meters | 0.5 | 0.18 |
| Element | | |
| Sun Sensors | 0.0 | 0.05 |
| Element | | |
| Magneto Torquers | 0.0 | 2.0 |

Figura B.2 - Página 1 de um exemplo de relatório gerado pelo SatBudgets antes da integração com SpaceESB



| | | |
|---|---|---|
| Element | | |
| CCSDS TC | 0.0 | 0.0 |
| Element | | |
| CCSDS TM | 0.0 | 0.0 |
| Element | | |
| Momentum Wheel | 2.4 | 0.3 |
| Element | | |
| Wiring ACDH | 0.0 | 0.0 |
| Element | | |
| Receiver 1 | 2.0 | 0.6 |
| Element | | |
| Transmitter 1 | 10.0 | 0.6 |
| Element | | |
| Receiver 2 | 2.0 | 0.6 |
| Element | | |
| Transmitter 2 | 10.0 | 0.6 |
| Element | | |
| Diplex 1 | 0.0 | 0.0 |
| Element | | |
| Diplex 2 | 0.0 | 0.0 |

Figura B.3 - Página 2 de um exemplo de relatório gerado pelo SatBudgets antes da integração com SpaceESB



| | | |
|---|---|---|
| Element | | |
| Hybrid | 6.0 | 2.0 |
| Element | | |
| S Band 1 | 0.0 | 0.2 |
| Element | | |
| S Band 2 | 0.0 | 0.2 |
| Element | | |
| Wiring TMTC | 0.0 | 0.4 |
| Element | | |
| Solar Panel | 0.0 | 2.3 |
| Element | | |
| Power Control and Dist. Unit | 10.0 | 4.0 |
| Element | | |
| Batter y | 0.5 | 3.5 |
| Element | | |
| Wiring PSS | 0.0 | 3.0 |
| Element | | |
| Experimental Payloads | 0.0 | 0.0 |
| Element | | |
| Experimental Digital DCS | 2.0 | 1.5 |

Figura B.4 - Página 3 de um exemplo de relatório gerado pelo SatBudgets antes da integração com SpaceESB

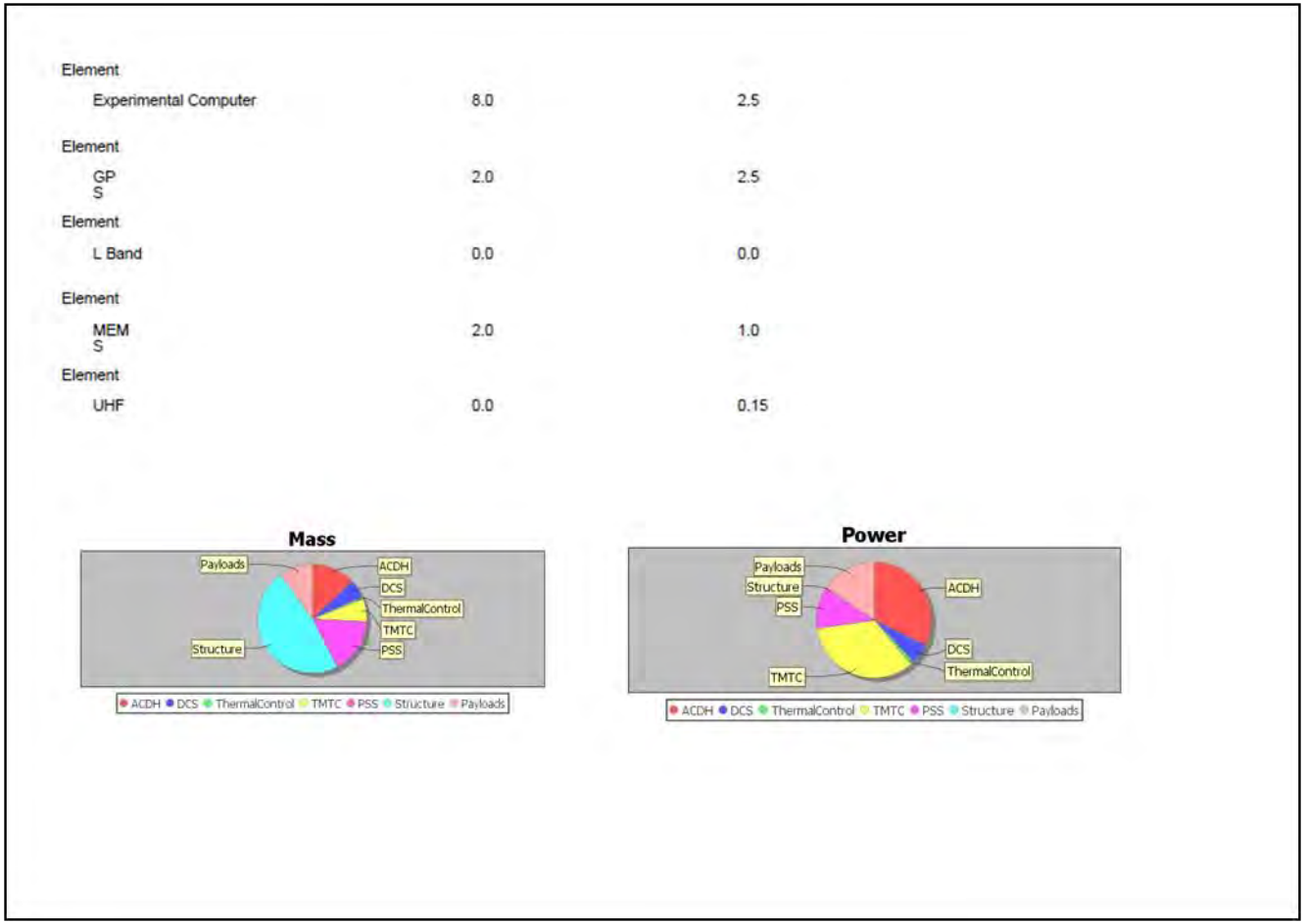



Figura B.5 - Página 4 de um exemplo de relatório gerado pelo SatBudgets antes da integração com SpaceESB

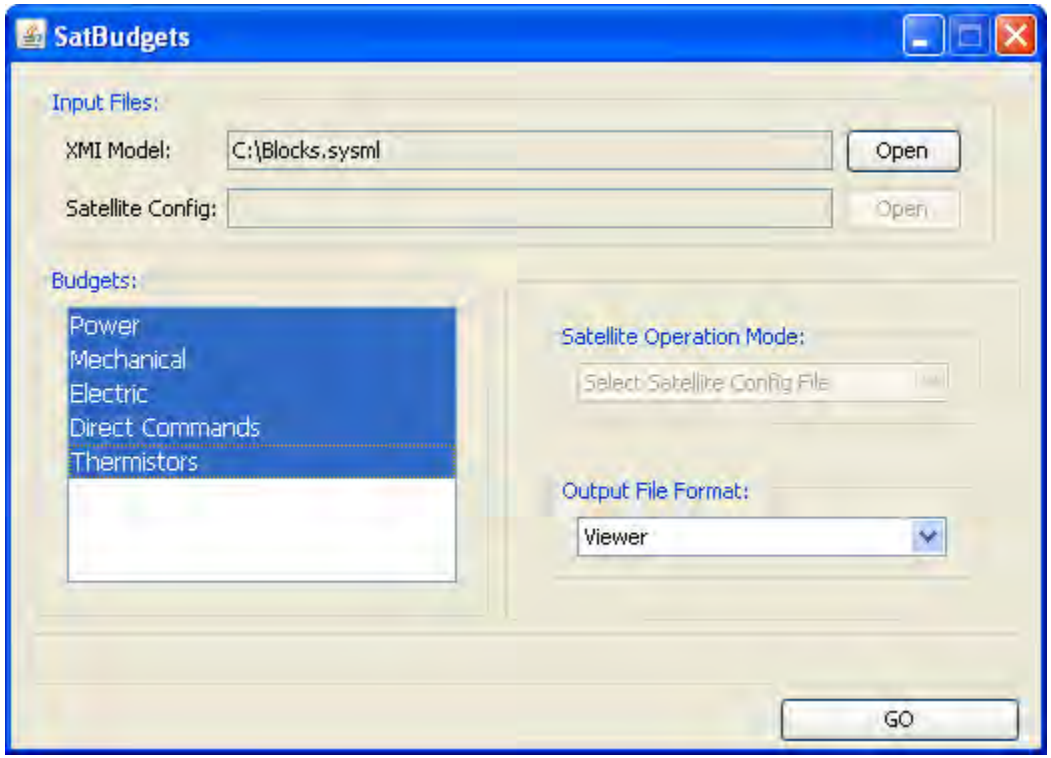



Figura A.6 - Tela principal do SatBudgets após a integração com SpaceESB

# SatBudgets Report

**Category: Blocks**

| Equipment | Active Power | Mass | Direct Commands | Thermistors |
|---|---|---|---|---|
| Structure | 0.0 | 36.6 | 4 | 4 |
| | 0.0 | 36.6 | 4.0 | 4.0 |

**Category: DCS**

| Equipment | Active Power | Mass | Direct Commands | Thermistors |
|---|---|---|---|---|
| Operational DCS | 5.0 | 3.0 | 2 | 2 |
| S Band 1 | 0.0 | 0.2 | 2 | 2 |
| UHF DCS | 0.0 | 0.15 | 2 | 2 |
| S Band 2 | 0.0 | 0.2 | 2 | 2 |
| Wiring DCS | 0.0 | 0.6 | 2 | 2 |
| | 5.0 | 4.15 | 10.0 | 10.0 |

**Category: ACDH**

| Equipment | Active Power | Mass | Direct Commands | Thermistors |
|---|---|---|---|---|
| OBC | 26.0 | 7.5 | 2 | 2 |
| Magneto Meters | 0.5 | 0.18 | 2 | 2 |
| Sun Sensors | 0.0 | 0.05 | 2 | 2 |
| Magneto Torquers | 0.0 | 2.0 | 6 | 6 |
| CCSDS TC | 0.0 | 0.0 | 2 | 2 |
| CCSDS TM | 0.0 | 0.0 | 2 | 2 |
| Momentum Wheel | 2.4 | 0.3 | 2 | 2 |
| Wiring ACDH | 0.0 | 0.0 | 6 | 6 |
| | 28.9 | 10.03 | 24.0 | 24.0 |

**Category: TMTC**

| Equipment | Active Power | Mass | Direct Commands | Thermistors |
|---|---|---|---|---|
| Receiver 1 | 2.0 | 0.6 | 2 | 2 |

Figura B.7 - Página 1 de um exemplo de relatório gerado pelo SatBudgets após integração com SpaceESB

| | | | | |
|---|---|---|---|---|
| Transmitter 1 | 10.0 | 0.6 | 2 | 2 |
| Receiver 2 | 2.0 | 0.6 | 2 | 2 |
| Transmitter 2 | 10.0 | 0.6 | 2 | 2 |
| Diplex 1 | 0.0 | 0.0 | 2 | 2 |
| Diplex 2 | 0.0 | 0.0 | 2 | 2 |
| Hybrid | 6.0 | 2.0 | 2 | 2 |
| S Band 1 | 0.0 | 0.2 | 2 | 2 |
| S Band 2 | 0.0 | 0.2 | 2 | 2 |
| Wiring TMTC | 0.0 | 0.4 | 2 | 2 |
| | 30.0 | 5.2 | 20.0 | 20.0 |
| **Category: PSS** | | | | |
| **Equipment** | **Active Power** | **Mass** | **Direct Commands** | **Thermistors** |
| Solar Panel | 0.0 | 2.3 | 2 | 2 |
| Power Control and Dist. Unit | 10.0 | 4.0 | 2 | 2 |
| Battery | 0.5 | 3.5 | 2 | 2 |
| Wiring PSS | 0.0 | 3.0 | 2 | 2 |
| | 10.5 | 12.8 | 8.0 | 8.0 |
| **Category: Payloads** | | | | |
| **Equipment** | **Active Power** | **Mass** | **Direct Commands** | **Thermistors** |
| Experimental Payloads | 0.0 | 0.0 | 2 | 2 |
| Experimental Digital DCS | 2.0 | 1.5 | 2 | 2 |
| Experimental Computer | 8.0 | 2.5 | 2 | 2 |
| GPS | 2.0 | 2.5 | 2 | 2 |
| L Band | 0.0 | 0.0 | 2 | 2 |
| MEMS | 2.0 | 1.0 | 2 | 2 |
| UHF | 0.0 | 0.15 | 2 | 2 |
| | 14.0 | 7.65 | 14.0 | 14.0 |

Figura B.8 - Página 2 de um exemplo de relatório gerado pelo SatBudgets após integração com SpaceESB


Figura B.9 - Página 3 de um exemplo de relatório gerado pelo SatBudgets após integração com SpaceESB

## ANEXO C - PUBLICAÇÕES EFETUADAS

### C.1 SpaceESB – A Proposal for an Enterprise Service Bus for Spacecraft Conceptual Design

Este trabalho, desenvolvido durante a pesquisa de Mestrado em Engenharia e Gerenciamento de Sistemas Espaciais em coautoria com o Prof. Dr. Walter Abrahão dos Santos, foi publicado no evento 21st Annual International Symposium que ocorreu no período de 20-23 de junho de 2011, em Denver, Colorado-EUA.

### C.2 Automating Services for Spacecraft Conceptual Design via an Enterprise Service Bus

Este trabalho, desenvolvido durante a pesquisa de Mestrado em Engenharia e Gerenciamento de Sistemas Espaciais em coautoria com o Prof. Dr. Walter Abrahão dos Santos, foi publicado no evento ISPE Concurrent Engineering 2011 - CE2011 que ocorreu no período de 4-8 de julho de 2011, em Massachusetts, EUA, no Massachusetts Institute of Technology.

### C.3 Automating Services for Spacecraft Conceptual Design via an Enterprise Service Bus

Este trabalho, desenvolvido durante a pesquisa de Mestrado em Engenharia e Gerenciamento de Sistemas Espaciais em coautoria com o Prof. Dr. Walter Abrahão dos Santos, foi publicado na forma de pôster no evento 2º Workshop em Engenharia e Tecnologias Espaciais que ocorreu no período de 3-4 de maio de 2011, São José dos Campos-SP, no Instituto Nacional de Pesquisas Espaciais-INPE.

## PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.